Traceability, Linkability and Policy Hiding in Attribute-Based Signature Schemes

El Kaafarani, Ali

Award date:
2015

Awarding institution:
University of Bath

Traceability, Linkability and Policy Hiding in Attribute-Based Signature Schemes

submitted by

Ali El Kaafarani

for the degree of Doctor of Philosophy

of the

University of Bath

Department of Computer Science

February 2015

Acknowledgements

I would like to express my special appreciation and thanks to my supervisor Professor James Davenport, this work would have been impossible without your help. I would like to thank you for encouraging my research and giving me the opportunity to work on this challenging topic. I also want to express my special thanks to Dr. Liqun Chen for being a brilliant supervisor during my internship at HP Labs. Thank you Li for all the time you dedicated to me, all your helpful explanations, comments and suggestions. I would like to also thank my second supervisor Dr Russell Bradford for all his help during my PhD.

I would especially like to thank Professor Fares Haddad and all of his clinic team in London, your help was tremendous.

Thanks to all people with whom I collaborated in the work presented in this thesis, James Davenport, Liqun Chen, Essam Ghadafi, and Dalia Khader. I learned a lot from these collaborations.

Thanks to all my colleagues at both Bath University and HP labs Bristol. My work experience in both environments was such an enjoyment.

I would like to thank my PhD examiners, Prof Guy McCusker and Dr Feng Hao, the Viva was a very special experience!

Special thanks to my mother, my family, and particularly to my uncle/friend for all their support, prayers, and encouragement. I would also like to thank all of my friends who supported me all the way and in all possible ways, both in Beirut and Bath.

To my better half, the piano girl Joe, you are the nice melody that always brings the peace to my mind and all the optimism of the world to our life. Thank you sweetest heart!

Abstract

Often we are less concerned with *who* signed something than with *what* attributes (director of this company etc.) they have. We propose three Attribute Based Signature schemes, namely, Decentralised Traceable Attribute Based Signatures DTABS, Attribute Based Signatures with User-Controlled Linkability ABS-UCL, and Attribute Based Signatures with Hidden Expressive Policy ABS-HEP. The *Traceability* assures that signatures in dispute, caused by any misuse/abuse cases, can be traced back to their signers. The judge of public opinion guarantees that no misattribution (framing) can take place. Additionally, *User-Controlled Linkability* gives a lightweight solution to session–style ABS; signers can *choose* to link some of their signatures that are directed to the same verifier, and the verifier will be convinced that those signatures are signed by the same anonymous person. *Hidden expressive policy* gives the organizations the flexibility to change their signing policies without notifying the outside.

All the three schemes are given and proven generically in a modular way. Instantiations for the first two schemes are also given to show both feasibility and practicality of the proposed schemes. The first two schemes substantially improve the state-of-the-art of Attribute Based Signatures that use Bilinear maps as a building block and shape it into a practical form, offering a *decentralised* version of ABS where multiple authorities are involved and yet no reliance on a central authority is needed.

In the third scheme, we move ABS into a new stage, where we increase the level of expressiveness of the signing policies to use general circuits, and at the same time, we give the signer the ability to fully hide his signing policy. This scheme makes use of hardness assumptions on the newly realised cryptographic building block, i.e. Multilinear maps.

Contents

List of Figures

Chapter 1

Introduction

This chapter is an an introduction to the thesis' contents. First, we give motivations and definitions to explain why the novel ideas achieved in this thesis are indeed useful and practical. No technical details are given in this chapter but rather a resumé of the contributions that we claim in the rest of it. After giving an explanation on where/how attribute based cryptography works, we present the three main technical contributions of this thesis which represent its main core:

- The first is *Decentralized Traceable Attribute Based Signatures* DTABS, *CT-RSA'14* version (published in [EKGK14], full version in [EKGK13]).
- The second is *Attribute Based Signatures with User-Controlled Linkability* ABS-UCL, *CANS'14* (published in [EKCGD14]).
- The third one is *Attribute Based Signatures with Hidden Expressive Policy* ABS-HEP [EKC14], US Patent PCT/US2014/047773, filed for Hewlet Packard (HP) company on 23/07/14.

Modularity plays an essential role in this thesis. Attribute based signatures, ABS, is among the hot and promising topics in applied cryptography. We aim at providing *modular generic solutions* to some open questions concerning ABS, e.g. decentralization, traceablity, user-controlled linkability, and hidden policy. These features are crucial to safely put ABS into practical use. Note that it's not always possible to give a generic solution for any crypto problem. The importance of the proposed solutions is that they are completely independent from concrete building blocks. To instantiate

a given generic scheme, one can employ the most efficient building blocks that fit the bill of the security requirements defined in the security model of the proposed scheme.

1.1 Why Attribute based Cryptography?

Access control It is very well known that access control appears in many aspects of our life. One can mention the physical access through security gates at a building, or virtual access by a computer program to a memory. Attribute/role based access control play a tremendous role in this area; for instance, someone who has got the right role or the right set of attributes can be granted access to that computer memory.

Confidentiality A trivial way to protect a confidential document is to encrypt it and hold the encryption keys. A coarse-grained access control is when you allow specific people (e.g. specified by their names) to access these confidential data, but this is definitely not the most efficient way. Here lies the importance of the attribute based encryption. Attribute-based encryption (ABE) is a cryptographic mechanism for enforcing fine-grained access control to plaintext, whether it is a secret PIN for opening a door, or a confidential document of a company. In another sense, it simultaneously offers a fine grained access control mechanism and the confidentiality by means of encryption. Examples can range from distributed network storage (or *storage in the cloud*), through *social networks* (with different portions of your profile and different categories of posts encrypted such that only your selected circles of friends can decrypt, possibly on a per-circle basis), to regulatory compliance where service providers want to prove that they indeed comply with certain regulations/clients requirements

Authenticity Attribute Based Signatures ABS, in their turn, can be always used as a mechanism for authentication. The ability to sign a message where the signature can be verified correctly, implies the possession of a certain corresponding set of credentials which satisfy a certain policy[1]. The identity of the signer as well as the subset of attributes that have been used to sign the message stay hidden from the verifier. An

[1] A policy is a Boolean combination between some credentials, e.g. $A \vee (B \wedge C)$.

extra level of anonymity might require hiding the relations between the credentials which could be used to sign certain messages, and this case can also be covered by ABS. To deal with this case, the signer should use a special type of ABS where he can hide everything related to his attributes including the policy which he signs with respect to.

1.2 How does Attribute Based Cryptography work?

In attribute based cryptography, every participant has a signing/decryption key, to sign/decrypt some documents/ciphertexts, based on the attributes that he holds. Let's take the example of an attribute based signature (ABS) and see how it works. In ABS, a signer who possesses a set of attributes from the relevant authority/authorities, can sign a message with any predicate that is satisfied by his attributes. The signature reveals no more than the fact that a single user, with some set of attributes satisfying the predicate, has attested to the message. In particular, the signature should hide the attributes used to satisfy the predicate and any identifying information about the signer (that could link multiple signatures as being from the same signer[1]). Furthermore, users should not be able to collude to pool their attributes together. Assume that a signer wants to use a subset of his attributes $\mathcal{A}_1 \subset \mathcal{A} = \{\alpha_i\}_{i=1}^n$ to sign a message w.r.t. a predicate Ψ. In the Signing algorithm, for a signer to be able to sign a message, he should have a set of attributes that satisfy the predicate Ψ, i.e. $\Psi(\mathcal{A}_1) = 1$. In Attribute Based Encryption, the scheme works in a similar way, i.e. for a user to be able to decrypt a ciphertext, he should have enough attributes to satisfy a certain predicate Ψ that has been used in the encryption algorithm, which means only specific people who have enough attributes/credentials can access this data that has been encrypted. There are two types of policies in terms of their monotonicity; a *monotone type* where there is no negation, e.g. $\Psi = (A \wedge B) \vee C$ and a *non-monotone type* where negation is allowed e.g. $\Psi = (A \wedge B) \vee \neg C$. Policies also differ from each other by their level of expressiveness as follows: There is the *threshold* type, where one can satisfy Ψ if he has n out of m attributes. There is also the *predicate* type where the fan-outs of the nodes are limited to 1, whereas the most expressive type of policies are the *general circuits*, where their fan-outs are greater than or equal to one.

[1] See chapter 5 for more details on user-controlled linkability.

1.3 Modular Approach to build ABS

There are two approaches to design cryptographic schemes; The first is the *modular* approach in which the *whole* is divided into manageable, feasible and secure smaller tasks which we usually call the *building blocks* of the scheme. One has to stick these building blocks together in a secure way in order to have the scheme in its final shape. The Non-Interactive Zero Knowledge Proof (NIZK) and its weaker version, the Non-Interactive Witness indistinguishable (NIWI), can be thought of as special modules because of their special role in *gluing* other modules together to finally yield secure cryptographic schemes.

The second approach, is mostly the industrial one or what is called *ad hoc* approach, therein, the main motivation is to get a practical/efficient scheme, regardless to how complex/non-standard the proof of the whole scheme could become. There is no doubt that the former approach has the great feature of making use of existing schemes, but the most important feature in the modular design is that the modules used to realise such a scheme can be replaced by more efficient/secure[1] versions that would serve better than the originals.

1.4 Decentralized traceable attribute based signatures

The first contribution that we present in this thesis is a modular approach to build Decentralized Traceable Attribute Based Signatures (DTABS) [EKGK14]. DTABS has been built on top of multiple secure modules. In order to get an efficient scheme based on cryptographically secure modules, the challenge is to find suitable schemes that fit the bill for both efficiency and security at the same time. In DTABS, as shown in Figure 1-1, we use One Time Signatures schemes (OTS), Unforgeable Digital Signatures, Bilinear Maps, CCA-Encryption Scheme, Span Programs, and NIZKs that represent the state of the art in terms of efficiency and at the same time are compatible with each other.

Intuition of DTABS As illustrated in Figure 1-2, a signer with identity id receives a set of attributes from *possibly* more than one attribute authority (AA), where the

[1] In the event that a weakness is discovered in one or more concrete modules, one can replace them by different modules that are more secure.

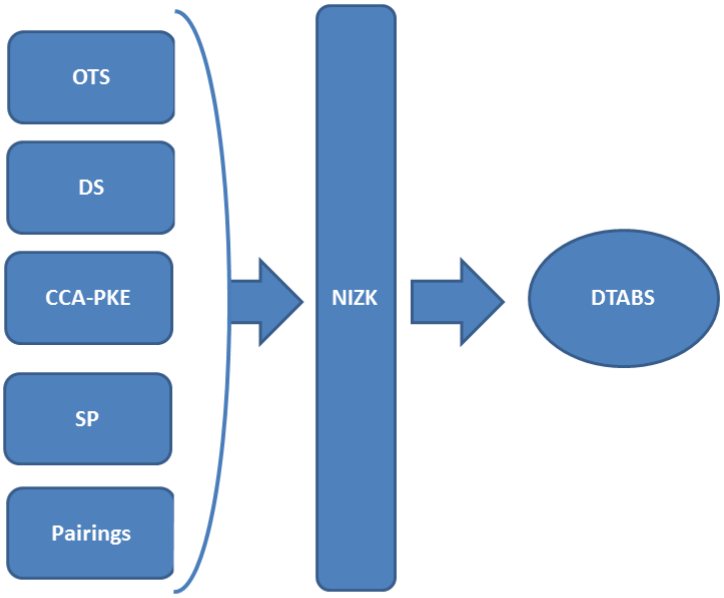


Figure 1-1: Modules connected together to get DTABS

attributes are in reality *signatures* on the concatenation of attributes names and his identity. To sign a message, a signer can use any subset of his attributes as long as it satisfies the signing predicate. He first needs to produce a NIZK proof that is formed of two parts; the first part is actually used to convince the signer that he can satisfy the predicate whereas in the second part, in order to provide the *traceability*, the signer id should encrypt his own identity id, and produce the second part of the NIZK that convinces the verifier that the id which has been encrypted is *the same* as the one in the signatures which he got from the attribute authorities. The opener (or tracer) has the secret key of the encryption scheme, which means that he can open any signature by simply decrypting the ciphertext attached to it. Moreover, the opener should also send the *Judge*[1] another NIZK (NIZK2) to convince him that he has correctly opened the signature (to prevent framing users).

[1]The Judge doesn't hold any secret keys and therefore one can think of it as *the judge of a public opinion.*

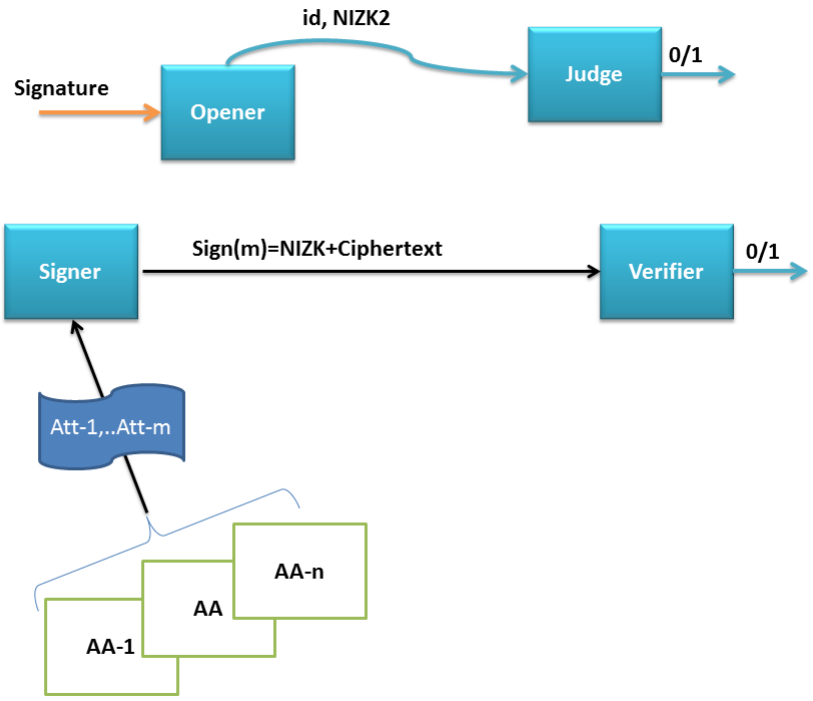


Figure 1-2: Communications between different DTABS entities

1.4.1 Contribution

DTABS **main contribution**

- Formal security model for decentralized traceable attribute-based signatures.
- Decentralization: Our focus is on the *more realistic* setting where there are multiple attribute authorities responsible for distributing the attributes/credentials to the users, and hence the word *Decentralized*.
- Traceability: This is a very important security requirement when it comes to real life. *Anonymity* is a good feature that ABS usually offer, but it should be controlled in the sense that it cannot be abused, e.g. framing users, misuse/abuse cases. To solve this problem, we add two entities to the ABS scheme. The first is the *Opener* who will be called to *open* the signatures in question in order to remove the anonymity and therefore be able to tell who has produced these signatures. Moreover, we have a *Judge* who has no secret keys, and yet he is able to tell whether or not the Opener has correctly opened the signature, thanks to the NIZK tool.

- Generic construction for DTABS: We give a security model for DTABS which implies the security requirements needed from the underlying modules that are going to be used in order to realise it, so anyone can actually instantiate the DTABS using different modules as long as they satisfy the security requirements stated in the generic construction.

- Moreover, we present two example instantiations of the generic construction and provide the first construction not relying on idealized assumptions. Our constructions meet strong security requirements and permit expressive signing policies. We also have a Random Oracle Model (ROM) instantiation that is presented in the following chapter for reasons of clarity.

1.5 ABS with User-Controlled Linkability

In chapter 5, the main concern that we take into consideration is the practicality. We propose a solution for a very important and practical issue when it comes to communication between people over the Internet. While there is no doubt that anonymity is definitely required and desirable in a lot of applications, in other applications more confidence between Internet communicators that they are surely talking to each other, in analogous way to cookies, is also required and desirable. To get the best of both worlds, we introduce yet another useful feature to ABS, namely *user-controlled linkability* (UCL), hence the name ABS-UCL. UCL permits different uses than those the tracing authorities can offer. The tracing authority is generally thought of as "for trouble-shooting" whereas UCL is intended to be built into normal use. For instance, in the world of attributes, assume that a signer wants to open a session with a recipient (in a analogous way to the idea of cookies which can provide state above the stateless HTTP). Additionally, the signer wants to maintain this session in a convincing way that he continues to be the same person whom the recipient is communicating with, not somebody else who has enough credentials to satisfy the same policy in question. Additionally, he wants to do so without revealing any extra information other than this fact. One can easily see that the tracing authority can't be much help here, whereas user-controlled linkability can let a signer link a set of his signatures by using a special type of digital signatures, a *Tag scheme*, to sign the recipient's name "recip", so that the recipient can actually verify that a certain set of signatures surely belong to the same

signer without actually knowing who that signer is. Note that a tracer cannot provide this service as he needs to open all those signatures before he can tell whether or not they belong to the same signer.

We build ABS-UCL in a modular fashion, and this requires a non-interactive zero knowledge proof (NIZK system), two Unforgeable Digital Signature Schemes (one of them is deterministic), Span Programs, and bilinear groups. We give security definitions and a generic construction of a ABS-UCL. We also give a *practical* instantiation of the scheme with the full details. Finally, we show how to change the practical ABS-UCL in order to get a new construction of our DTABS that is much more efficient than those realised in the previous chapter, but now realised in the Random Oracle Model (ROM) [BR93].

1.5.1 Contribution

ABS-UCL **main contribution**

- Formal security model for attribute based signatures with user controlled linkability.
- Decentralization: Again, and similar to our DTABS, we deal with the case where there are multiple attribute authorities that are responsible for distributing the attributes/credentials to the users.
- User-Controlled Linkability: It allows for a signer to convince a certain verifier that a set of signatures all belong to him without revealing any further information about his identity or the attributes that he holds.
- Generic construction for ABS-UCL: We abstractly use the underlying modules needed to realise ABS-UCL. Security proofs are also given for the generic construction.
- Instantiations: We give a practical instantiation of ABS-UCL in the random oracle model. Moreover, we use the given instantiation to give a practical instantiation of our DTABS.

1.6 ABS with Hidden Expressive Policy

In Chapter 6, we introduce yet another novelty by presenting the first attribute based signature scheme that deals with the most expressive types of policies, i.e. circuits. Moreover, and in order to increase the level of the anonymity in the proposed scheme, the signer can actually hide the policy under which he signs the messages. This property can be very helpful in many real business cases. We present below one of the cases in which fully anonymous ABS for circuits can perfectly fit in.

Real Case Scenario Let us take a company as an example, although the proposed solution is suitable for various types of organizations. Assume that the company has employees in different positions. According to their positions, the employees are allowed to sign some messages on behalf of the company, e.g., the CEO alone can create a signature, a certain number of managers in certain levels can work together to make a signature, and ordinary employees can go through a referendum to generate a signature. These signatures should be indistinguishable outside the company. More formally, we want the identities of the signers, along with the internal hierarchy of the company, to be hidden from a signature verifier, as this is the sensitive information of the company. The only information that the verifier could learn from a given signature is that the signature was generated following company policies, without knowing anything about those policies.

1.6.1 Contribution

All the previous Attribute Based Signatures used bilinear maps as a building block to construct their schemes. In our scheme, we use multilinear maps that allow us to overcome some of the limitations caused by the bilinear maps, namely, the type of the policy. In our scheme, it is possible to go from boolean formulas/span programs (circuits with fanout=1) to deal with general circuits (fanout $\geq$ 1 see Fig. 1-3), and yet avoid the *backtracking* attack [GGH$^+$13b] that could happen at any OR-gate if one used Bilinear maps (for more details on the attack see Section 6.2). This is all possible thanks to the multilinear maps.

The contribution of Chapter 6 is three-fold, as follows:

- *Circuits*: It is the first attribute-based signature scheme that can deal with general circuits as the signing policy. All previous schemes were restricted to the

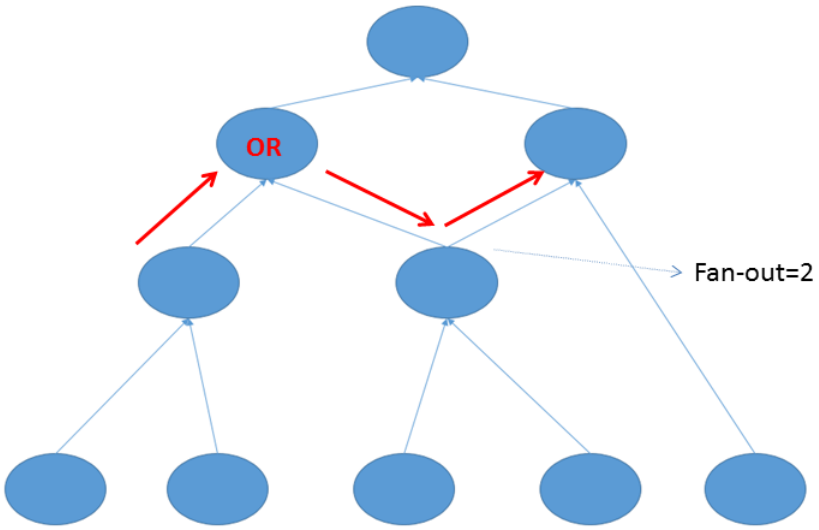


Figure 1-3: Example of a circuit

predicate type due to their reliance on bilinear maps.

- *Policy privacy*: The proposed attribute-based signature scheme allows the signing policy to be hidden, which means that signers are not asked to reveal any sensitive information about their roles or about the internal hierarchies of their companies.

- *Constant size signature*: The proposed scheme provides a constant size signature, i.e. independent of the number of attributes, and the verification process is very efficient.

1.7 Conclusion

In this chapter, we outlined the main contributions in this thesis. We gave an introduction to each of the three Attribute Based Signature schemes which we propose, namely, Decentralized Traceable Attribute Based Signatures (DTABS), Attribute Based Signatures with User-Controlled Linkability (ABS-UCL), and Attribute Based Signatures with Hidden Expressive Policy (ABS-HEP). Each of these schemes aims at solving essential security problems (e.g. decentralization, traceability, user-controlled linkability, hidden policy).

Chapter 2

Cryptographic background

2.1 Introduction

Cryptography was originally the art of concealing information. More precisely, rewriting comprehensible information in an incomprehensible format that makes it unreadable without the possession of the "secret key". Modern cryptography, is the science that deals with problems related to information security, e.g. confidentiality, authenticity, data integrity, etc. In this chapter, we will give an introduction to some important cryptographic notions, symmetric/asymmetric encryption schemes and their related security notions and different types of signature schemes and their related security notions. Moreover, we talk about an important topic in modern cryptography, *provable security*, by which we can show that certain cryptosystems are secure assuming that certain mathematical problems are computationally hard (e.g. factoring, discrete logarithm, etc.). We also compare the two famous security models in cryptography, the Random Oracle model and the Standard model. In the former, we assume the existence of an ideal hash function, that would be eventually replaced by "regular secure hash functions" when implemented, whereas in the latter, we don't rely on idealized assumptions at the expense of believing that some "non–standard" assumptions hold. Finally, we give an introduction to attribute based cryptography, show the progress in this interesting topic starting from 1984 where Identity based encryption was introduced. Furthermore, we talk about different types of both attribute based encryption (ABE) and attribute based signatures (ABS).

2.2 Encryption and Related Security Definitions

2.2.1 Symmetric Encryption

The symmetric encryption setting considers two parties who share a secret key and use it to encrypt/decrypt the transmitted data between them. A symmetric encryption scheme consists of three algorithms that are $(\mathsf{E.KG}, \mathsf{E.Enc}, \mathsf{E.Dec})$; let $\mathcal{M}$ be the message space, $\mathcal{C}$ the space of ciphertexts, and the key space is $\mathcal{K}$. Below are the descriptions of the algorithms:

- $\mathsf{E.KG}(1^\lambda)$: is a randomized algorithm that, given the security parameter λ[1], returns a key $\mathsf{sk} \in \mathcal{K}$.
- $\mathsf{E.Enc}(\mathsf{sk}, m)$: is a randomized or stateful algorithm that on input of a key $\mathsf{sk} \in \mathcal{K}$ and a paintext $m \in \mathcal{M}$, outputs a ciphertext $c \in \mathcal{C}$.
- $\mathsf{E.Dec}(\mathsf{sk}, c)$: is a deterministic algorithm that on input of a key sk and a ciphertext $c \in \mathcal{C}$ outputs a message $m \in \mathcal{M} \cup \perp$.

Correctness: The encryption scheme is *correct* if the receiver can always get the original plaintext when he decrypts a given ciphertext, more formally:

$$\forall m \in \mathcal{M}, \Pr[\mathsf{sk} \leftarrow \mathsf{E.KG}(1^\lambda) : \mathsf{E.Dec}(\mathsf{E.Enc}(\mathsf{sk}, m), \mathsf{sk}) = m] = 1$$

The main problem with the symmetric key encryption is sharing the key between the parties over insecure channels. In [DH76], they define the *public key* encryption notion for the first time in the public literature, where two parties can communicate over insecure channels, to *securely* share a key. It's computationally infeasible for an adversary who is listening to their communication, to recover the key from the public information which they transmit back and forth between each other.

2.2.2 Asymmetric or Public Key Encryption

An asymmetric encryption scheme consists of the following algorithms:

[1]The security parameter is usually used to tune the security level of a cryptosystem to make breaking the scheme computationally infeasible. The higher the value of λ, the smaller the adversary's advantage in breaking the scheme. For instance, in an encryption scheme, the security parameter controls the length of the keys and ciphertexts.

- $\mathsf{E.KG}(1^\lambda)$: is a randomized algorithm that takes the security parameters as input and returns a pair of keys $(\mathsf{pk_e}, \mathsf{sk_e}) \in \mathcal{K}^2$, the public key $\mathsf{pk_e}$ and its matching secret key $\mathsf{sk_e}$, respectively.

- $\mathsf{E.Enc}(\mathsf{pk_e}, m)$: A randomized algorithm that takes a public key $\mathsf{pk_e}$, a plaintext $m \in \mathcal{M}$ and returns a ciphertext $c \in \mathcal{C}$.

- $\mathsf{E.Dec}(\mathsf{sk_e}, c)$: A deterministic algorithm that takes the secret key $\mathsf{sk_e}$ and a ciphertext $c \in \mathcal{C}$, and returns a message $m \in \mathcal{M} \cup \perp$.

Correctness: The encryption scheme is *correct* if the receiver can always get the original plaintext when he decrypts a given ciphertext, more formally:

$$\forall m \in \mathcal{M}, \Pr[(\mathsf{sk_e}, \mathsf{pk_e}) \leftarrow \mathsf{E.KG}(1^\lambda) : \mathsf{E.Dec}(\mathsf{E.Enc}(\mathsf{pk_e}, m), \mathsf{sk_e}) = m] = 1$$

2.2.3 Security Notions: CPA-CMA-CCA-AE

We clearly want the attacker to be unable to recover the secret key, or recover all of the plaintext. In 1949, Shannon [Sha49] gives the first precise security notion:

"The ciphertext should reveal no information about the plaintext"

Perfect Secrecy

More formally, Shannon defined perfect secrecy as follows:

Definition 1. *[Sma13] A cryptosystem has perfect secrecy if*

$$Pr(Plaintext = m | Ciphertext = c) = Pr(Plaintext = m)$$

for all plaintexts $m \in \mathcal{M}$ *and all ciphertexts* $c \in \mathcal{C}$.

Another way to define perfect secrecy is to say that the two distributions $\mathsf{E.Enc}(k, m_0)$ and $\mathsf{E.Enc}(k, m_1)$ are computationally indistinguishable for *any* two messages m_0 and m_1 (denoted $\mathsf{E.Enc}(k, m_0) \approx_p \mathsf{E.Enc}(k, m_1)$) This security notion is considered to be strong and very hard to achieve.

Semantic Security or Security against Chosen Plaintext Attacks CPA [KY00, GM84]

It says that $\mathsf{E.Enc}(k, m_0) \approx_p \mathsf{E.Enc}(k, m_1)$ should only hold for pairs that the attacker can exhibit and not for all pairs of messages in $\mathcal{M}$. Adversaries here are considered to be eavesdroppers, that just listen to the network traffic but they don't change or inject any packets. This definition cannot guarantee secrecy or integrity under active attacks. Note that any system that is secure against CPA is considered to be offering confidentiality.

Integrity or Security against Chosen Message Attack CMA [KL07]

Integrity (or *data-origin authentication*) is to prevent an adversary, or a "forger", from creating a message that looks like it comes from a legitimate sender. To do this, the sender adds a tag to the message. The receiver then verifies that this tag belongs to this message, using a shared key. More formally, a system is secure against chosen message attack if no efficient adversary is able to produce a tag to any message that can be verified correctly by the receiver. He should not be able to produce a valid tag for a message even if he already knows another tag for it. The basic way to ensure message integrity is to apply MAC techniques [PP10] to add tags to the messages. The security against chosen message attack doesn't guarantee confidentiality, that's why they came up with a more general and natural notion of security which is the security against chosen ciphertext attack (CCA). To clearly explain the idea of the CCA security notion, we will first define two important notions; Indistinguishability and Non-malleability.

Indistinguishability

This is based on the following game/experiment [RS92]: the adversary sends two messages of his choice to the challenger, and then gets an encryption of one of them. The adversary's goal is to tell which one was encrypted. The indistinguishability property intuitively implies that an adversary seeing only a ciphertext should not be able to extract the plaintext.

Non-malleability

Originally formulated by Dolev, Dwork and Naor [DDN91], this asks that an adversary who sees a ciphertext of one of several messages of his choice cannot come up

with further ciphertexts whose messages are "meaningfully" related to his challenge ciphertext, even if he cannot decrypt any of these himself.

Both Indistinguishability and Non-malleability are formulated by a heirarchy of four games, known as IND–CPA, IND–CCA1, Parallel CCA and IND–CCA2 respectively and which differ in when the adversary can additionally obtain decryptions of ciphertexts:

- CPA: the adversary never decrypts,
- CCA1: also known as *Lunchtime* attack. The adversary can *only* decrypt before he gets his ciphertext
- Parallel CCA: the adversary can decrypt adaptively before he gets his challenge ciphertext. After he gets his ciphertext he can just decrypt (once) as many messages as he wants but not adaptively.
- CCA2: the adversary can decrypt at any time, provided he does not ask for the challenge itself to be decrypted.

There are two types of each of the above attack types; the Non-malleable CCA attack (NM–CCA1) and the Indistinguishable CCA1 attack (IND–CCA1). The attacker's goal differs from one to the other. As we defined them above, in the NM–XXX, the cryptosystem is said to be *broken* if the attacker succeeds in making a new valid cipher out of another one. For the IND–XXX, the attacker is asked to distinguish between two ciphers with a non negligible probability. So what are the relations between these different attacks?

Let $\mathcal{F}$ be the attacker. The following table summarizes for each attack, the attacker's power and goal. Then, we will give the relationships between NM–XXX and the IND–XXX.

$\mathcal{F}$**'s power in** $1st$ **stage**	**Challenge**	$\mathcal{F}$**'s power in** 2^{nd}**stage**	**Guess**	**Attack's Name**
Enc		—		CPA
Enc/Dec		—		Lunchtime CCA (CCA1)
Enc/Dec		Non–adaptive Dec		Parallel CCA
Enc/Dec		Dec		Adaptive CCA (CCA2)

In each attack, the goal of the attackers could be one of two choices, Malleability or Distinguishability. Thus the system is said to be Non-malleable secure (NM) or Indistinguishable (IND) secure against any of the following attacks: CPA, Lunchtime CCA (CCA1), Parallel CCA, or Adaptive CCA (CCA2).

Relations among Security Notions

Malleability is considered as a stronger goal than the Distinguishability, therefore there will be a trade-off in any Goal-Attack relation. It can be illustrated by the following relationships [BS99, DDN91]:

- NM–XXX $\Rightarrow$ IND–XXX
- NM–XXX $\Leftrightarrow$ IND–(XXX $\cup$ Non-adaptive decryption queries in $2nd$ phase), where XXX $\in \{$CPA, CCA1, CCA2$\}$.

Replacing XXX by each member of the set would give us the following results:

- NM–CPA $\Leftrightarrow$ IND–Parallel CCA
- NM–CCA2 $\Leftrightarrow$ IND–CCA2 [1]
- NM–CCA1 $\Leftrightarrow$ IND–(Parallel CCA)

We will formalize the most interesting case of the security requirements for any encryption scheme, i.e. IND–CCA2 security [2] Besides the usual correctness requirement, IND–CCA is defined by the game in Fig. 2-1.

We say that an encryption scheme is IND-CCA secure if for all $\lambda \in \mathbb{N}$, all polynomial-time adversaries $\mathcal{F}$ have a negligible [3] advantage

$$\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{PKE},\mathcal{F}}(\lambda) = \left|\Pr[\mathsf{Exp}^{\text{IND-CCA-0}}_{\mathsf{PKE},\mathcal{F}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\text{IND-CCA-1}}_{\mathsf{PKE},\mathcal{F}}(\lambda) = 1]\right|$$

In other words, an adversary that chooses 2 messages, and receives the encryption of one of them, is not able to guess which message has been encrypted, even if it is able to ask for decryption of any ciphertext of its choice (except the challenge ciphertext).

[1] That's why sometimes they don't differentiate between the two notions NM and IND since CCA2 is the most used one.

[2] We will follow the literature and call it IND-CCA.

[3] In theory, the term negligible refers to a function $\epsilon(n)$ which is smaller than $1/n^{\alpha}$ for all $\alpha > 0$ and sufficiently large n. In practice, one can think of $\epsilon \leq 1/2^{80}$.

Experiment: $\mathsf{Exp}^{\text{IND-CCA-b}}_{\mathsf{PKE},\mathcal{F}}(\lambda)$:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- $(m_0, m_1, \mathsf{st}_{\mathsf{find}}) \leftarrow \mathcal{F}_{\mathsf{find}}\,(\mathsf{pk} : \mathsf{Dec}(\mathsf{sk}, \cdot))$, where $|m_0| = |m_1|$.
- $C_b \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$.
- $b^* \leftarrow \mathcal{F}_{\mathsf{guess}}\left(\mathsf{st}_{\mathsf{find}}, C_b : \mathsf{Dec}^{C_b}(\mathsf{sk}, \cdot)\right)$, where Dec^{C_b} returns $\perp$ if queried on C_b.
- Return b^*.

Figure 2-1: IND–CCA security game for PKE

Authenticated Encryption (AE) Authenticated encryption[1] (Fig. 2.2.3) is the result of combining both concepts, security against chosen plaintext attacks and security against chosen message attacks [BCK96, BN00]. It's a very natural notion of security where the adversary has *real power* to encrypt/decrypt messages of his choice. If a cryptosystem is secure against CPA and CMA then it is secure against CCA [BN00], hence AE $\implies$ (NM/IND)-CCA. To give an example of a good combination between MAC and CPA to guarantee the security against CCA attacks, one can recall the "Encrypt than MAC" method. Let the ciphertext be $c = (c_1, \mathsf{tag})$ where $c_1 := \mathsf{E.Enc}(\mathsf{sk}_{\text{enc}}, m)$ and $\mathsf{tag} = \mathsf{Sign}(\mathsf{sk}_{\text{mac}}, c_1)$. Since the Adversary cannot create a valid new tag, then he cannot create a valid cipher, so the decryption power will be useless (it always returns $\perp$ for invalid ciphers), and therefore the problem will be reduced to a normal chosen plain text attack which we already know the system is secure against.

Although authenticated encryption ensures confidentiality against active adversaries that have the powers to decrypt some ciphers, it doesn't prevent replay attacks or side channel attacks[2] on its own. TLS[3] is a real example of authenticated encryption

[1] In 2012, they started a competition to build new AE schemes. " CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness" [cae12].

[2] Like a timing attack or any other attack based on information gained from the physical implementation of a cryptosystem and not from brute forcing or theoretical weaknesses in the algorithms.

[3] Surprisingly, TLS doesn't apply the *Encrypt than MAC* technique but rather its opposite. It first applies a MAC to the plaintext, then adds up to 255 bytes of padding to get the message up to a multiple of the cipher (e.g., AES's) block size. Now it CBC-encrypts the record! Therefore the padding is not protected by the MAC, and hence the existence of padding oracle attacks.

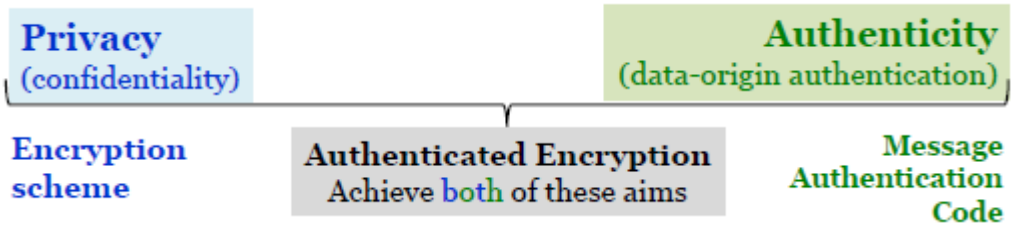


Figure 2-2: Authenticated Encryption

(TLS uses, amongst many other choices, CBC AES-128 for encryption and HMAC-SHA1 for MAC)[1]

2.3 Digital Signatures and Related Security Definitions

In 1976, Diffie and Hellman [DH76] introduced the concept of digital signatures. In this concept, a signature scheme consists of three algorithms:

- $\mathsf{KeyGen}(1^\lambda)$: takes as input a security parameter 1^λ, and outputs for a signer, a pair of a secret signing key and public verification key $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{Sign}(m, \mathsf{sk})$: takes as input a message m and the signer secret key sk, and outputs a signature σ.
- $\mathsf{Verify}(\sigma, m, \mathsf{pk})$: it outputs 1 if signature verifies correctly and 0 otherwise.

2.3.1 Security Definitions

- **(Existential) Unforgeability:[GMR88]** Unforgeability under adaptive chosen-message attack requires that any probabilistic polynomial time (PPT) adversary $\mathcal{F}$ that is given a signing oracle $\mathsf{Sign}(\mathsf{sk}, \cdot, \cdot)$ has a negligible advantage in winning the following game:
  - A key pair $(\mathsf{sk}, \mathsf{vk})$ is generated and vk is sent to $\mathcal{F}$.
  - $\mathcal{F}$ makes a polynomial number of queries to $\mathsf{Sign}(\mathsf{sk}, \cdot, \cdot)$.

[1] See [AP13] for a good explanation on current TLS/DTLS implementations' vulnerabilities, especially when it comes to side channel attacks.

– Eventually, $\mathcal{F}$ halts by outputting a tuple (σ^*, m^*) and wins if σ^* is valid on m^* and m^* was never queried to Sign.

- **Strong Ungforgeablity:[BSW06]** A signature system is said to be *strongly unforgeable* if the signature is existentially unforgeable and, given signatures on a certain message m, the adversary cannot produce a new signature on m.

- **One Time Signature (OTS)** A signature system is said to be an *unforgeable one time signature* if in the unforgeability game the adversary is restricted to a single signing query. A strongly unforgeable OTS is an unforgeable OTS system where the adversary can't produce a new signature on the queried message m.

2.3.2 Different Types of Digital Signatures

To give more flexibility to the signature scheme so it can be used in a wider range of real scenarios, they have come up with the idea of parametrizing the signing and verification keys [CH91, MPR11, RST01]. One can think of the verification key as VK$= (\Psi, \mathsf{pk})$ and the secret key as SK$= (w, \mathsf{sk})$. In the signing algorithm, there is the extra condition that $\Psi(w)$ must hold. For instance, given a signature σ on a message m generated by the signing key SK=(w, sk), σ verifies correctly against the verification key VK=(ψ, pk), if and only if w satisfies Ψ, i.e. $\Psi(w)$ holds. In some scenarios, where one might aim to preserve the privacy of signers, a signature generated by a user holding (sk, w) should reveal no information about w except the fact that it does satisfy the predicate Ψ. Different types/forms of Ψ would lead to different variants of digital signatures. Below, we try to briefly describe some of these variants.

Group Signatures Group signatures were first introduced by Chaum and Heyst in [CH91]. They defined as follows: A group of people wants to create a signature scheme, which we will call a *group signature scheme*, in which only members of the group can sign messages, the receiver can verify that it is a valid signature of that group (using the group verification key), yet the signer stays anonymous to the verifier, we call this property *anonymity*. In case of dispute later on, the signature can be "opened" to reveal the identity of the signer, this property is usually called *traceability*. In some systems there is a third party that can trace the signature, or undo its anonymity, using a special trapdoor (tracing key). Other systems might support *revocation* where

the group manager can selectively disable the membership of a specific member but without affecting signature scheme. One can think of the predicate Ψ here as a membership test to tell whether or not a signer belongs to a given group. Two security models/schemes were defined, the BMW model [BMW03] where the security of such group signature can be achieved by proving Full–traceability and Full–anonymity, and the BSZ one [Men05] in which the security requirements are Anonymity, Traceability and Non-frameability (the reason to have different security requirements is the change in the level of trust in Issuer and Opener authorities). In the BMW model, the group is considered to be static whereas in the BSZ model, the group is dynamic in the sense that it contains a Join protocol, which means that members can join at any time. The signatures produced by different group members look indistinguishable to their verifiers, whereas the group manager has the power to reveal the identity of misbehaving signers.

Ring Signatures As defined in [RST01], a *ring signature* makes it possible to specify a set of possible signers without revealing which member actually produced the signature. Unlike group signatures, ring signatures have no group managers, no setup procedures, no revocation procedures, and no coordination. Any user can choose any set of possible signers that includes himself, and sign any message by using his secret key and the others' public keys, without even getting their approval or assistance. Ring signature schemes are simplified group signature schemes which have no managers. In [RST01] they called such signatures "ring signatures" instead of "group signatures" since rings are geometric regions with uniform periphery and no center. A ring here simply represents a set of possible signers.

2.4 Provable Security

Roughly speaking, a cryptographic scheme is *provably secure* if it's secure under some specified mathematical hardness assumptions. We assume that a certain problem is computationally hard, and we prove that any efficient algorithm that one might find to break the given scheme can be used as an efficient algorithm (oracle) to solve the underlying "hard" problem. Herein lies the role of the mathematical problems that are believed to be hard, e.g. Discrete Logarithm problem (Dlog), Diffie–Hellman problem, Factoring large composite numbers, Finding the shortest vector in lattices, RSA

assumption and many others. For example, we might say that a certain scheme S is secure under the assumption that the Computational Diffie–Hellman CDH problem is hard. We then say that the security of the scheme S can be "reduced" to the CDH problem. Therefore, breaking the scheme leads to solving the CDH problem. Below, we define the game-based model of security proofs in cryptography, which happens between a challenger and an adversary.

2.4.1 Challenger $\mathcal{C}$ vs Adversary $\mathcal{F}$

Let us first state some facts about the challenger $\mathcal{C}$ and the adversary $\mathcal{F}$ in a general game-based security proof:

- X: A mathematical problem that is believed to be hard.
- Challenger's goal: $\mathcal{C}$ wants to solve problem X.
- Adversary's power: $\mathcal{F}$ knows how to break the Scheme S.

Description: We assume that there exists an adversary $\mathcal{F}$ that has enough power to efficiently[1] break the scheme S. The challenger will manipulate the adversary $\mathcal{F}$ to let her help him in solving his problem X. The challenger $\mathcal{C}$ will simulate the behaviour of the scheme S to convince the adversary that she is "communicating" with the scheme itself. It means that $\mathcal{C}$ has to efficiently respond to all of $\mathcal{F}$'s queries that S is supposed to answer. The challenger $\mathcal{C}$ will always start from the given problem instance that he had at the beginning. $\mathcal{C}$ will involve these given in all his responses to $\mathcal{F}$ so at the end when $\mathcal{F}$ breaks the system S, $\mathcal{C}$ will use this information/result in order to solve the problem X. We can formulate what we have just said above in the following statements.

- If X is hard, then S is provably secure.
- If there exists an Adversary $\mathcal{F}$ who can efficiently break S, then there exists a challenger $\mathcal{C}$ who uses $\mathcal{F}$'s power to efficiently solve X.

[1] In a (polynomial) time that is better than brute forcing.

2.4.2 Random Oracle Model (ROM) vs Standard Model

Hash functions are very important tools in cryptography. It's not easy to imagine their absence as most of the existing cryptosystems rely on them. By definition, hash functions transform a long input into a fixed-size output, that we usually call message digest. A *good* hash function must satisfy certain well known properties; one way, i.e. hard to invert, collision resistant, i.e. hard to find two different messages that hash down into the same digest, second pre-image resistance, i.e. given a message m_1, it is difficult to find another message m_2, where $m_2 \neq m_1$, but both hash to the same digest. The fact that the range is smaller the the domain means there must be collisions but they should be hard to find, hence finding collisions is possible but the probability of finding one should be negligible.

The essential question that determines the quality of a given hash function is to test the *randomness* of its outputs. Do perfectly (ideal) random hash functions exist? In practice the answer is definitely no, logically speaking, it's going to be "too slow to compute, too big to store"! In theory, ideal hash functions are called *random oracles* [BR93]. An ideal hash function has all the good properties of an excellent hash function in addition to its *exclusive* property which is the randomness. Any cryptosystem that is proven secure using a random oracle as its hash function is said to be provably secure in the random oracle model ROM, whereas cryptosystems that *only* use regular (non-ideal) hash functions in their proofs are considered to be provably secure in the standard model. Most of the implemented schemes that are in use nowadays are proved secure in the ROM, e.g. RSA-OAEP is one of them, which has been proven provably secure in the ROM reduced to the hardness of the RSA problem. What is the difference between the random oracle model and the standard model (where we use the a real hash function)? Is it just the randomness that is offered by the random oracle? According to [Gal12], in the proofs, wherever we use the random oracle, we assume that the random oracle is a third party accessible by all other parties including the adversary. Which is logical somehow, because we can't once say that the adversary can't break the system due to its limited power and then after that say that she can compute a random function! In this case, the whole scenario is not valid any more, and the adversary can simply brute force the scheme. So the random oracle is an "independent" party that answers all the queries of all other parties[1].

[1]For more details on the comparison between Random Oracle model and Standard Model, we refer the reader to *The Random Oracle Model: A Twenty-Year Retrospective* [KM15].

2.5 Attribute Based Cryptography

Attribute-based cryptography has emerged as an important research topic in recent years. It offers a versatile solution for designing role-based cryptosystems. In such systems, users are assigned attributes, and private operations (e.g. decryption/signing) are associated with security policies. Only users possessing attributes satisfying the policy in question can perform such operations. For instance, "any registered doctor can sign a prescription", or "any director of the company can access its encrypted data", etc. The first proposals of attribute-based cryptosystems were: an encryption scheme by Goyal et al. [GPSW06] (inspired by Sahai and Waters [SW05]) and a signature scheme by Maji et al. [MPR08]. Moreover, Attribute based cryptography can be considered as a natural extension and generalisation of the work that have been done before in public key cryptography field. We will give an overview of progression in this field, starting from Identity Based Encryption (IBE), leading to Attribute Based Encryption (ABE) and Signatures (ABS).

2.5.1 Identity Based Encryption (IBE)

The concept of identity-based encryption (IBE) is due to Shamir in 1984 [Sha84]. The idea of it is to allow a party to encrypt a message using the recipient's identity as a public key. A trusted/central authority is responsible for providing private keys in correspondence with every public key. The main advantage of IBE over traditional public-key encryption is by simplifying the implementation of the secure communication between the users by avoiding certificate management. The users can simply use their email addresses as their identities.

In 2001, Boneh and Franklin [BF03] found an elegant solution to Shamir's idea using pairings (See 3.5). Separately, Cocks [Coc01] in the same year also found another implementation of the IBE system based on quadratic residues, which is the only IBE that doesn't use pairings. Another identity based encryption based on pairings has been proposed by Sakai-Ohgishi-Kasahara, 2000.

One can say that Boneh-Franklin's IBE was the principal work that has catalysed a series of further research in this domain. Their original scheme is provably secure in the random oracle model based on the hardness of the Decisional Bilinear Diffie-Hellman using Type-1 pairings. Subsequently, there were so many attractive areas of research to improve Boneh-Franklin scheme, like proving it secure without random

oracle, extending it to Hierarchical Identity based encryption (HIBE), upgrading it to use Type-2 or Type-3 pairings and much more. Galindo in [Gal05] has corrected a flaw in the security reduction of the original BF-IBE scheme, and he upgraded it to Type-2 pairings assuming the existence of a hash function to $\mathbb{G}_2$ which doesn't seem to easily exist as pointed out by Chatterjee-Menezes in [CM09]. Canetti *et al.* [CHK03] and separately, Boneh-Boyen [BB04b] in 2004 introduced the Selective-ID [1] secure identity based encryption that has been proven CCA secure without random oracle, and another one which was considered quite impractical, which is Adaptive-ID CCA secure [BB04c].

In 2004, based on Boneh-Boyen work, Waters [Wat04] came up with an IBE which is Adaptive-ID CCA secure with relatively long public parameters, it has been followed up by some works done by Naccache [Nac05] and separately by Chatterjee-Sarkar [CS06] that have showed the trade-off between keeping the same length of the public parameters of Waters' scheme, or decreasing them by loosening the security reduction. In 2006, Gentry introduced an IBE scheme that is Adaptive-ID CCA secure [Gen06], has a tight security reduction, and efficient. But the security proof was based on a non-static assumption, which means that the number of the elements in the instance of the hard problem depends on the number of the queries the attacker makes. Note that Gentry's security proof is based on the idea that has been used in the security proof of Cramer-Shoup's public key cryptosystem in 1998 [CS98].

2.5.2 Attribute Based Encryption (ABE)

Attribute based encryption can be seen as a generalisation of the identity based encryption technique. It offers a fine-grained access control over encrypted data. The data, instead of being encrypted under someone's public key (e.g. public key cryptosystem) or someone's identity (or email address, etc. like IBE), it will be encrypted with respect to a certain policy or it will embed some attributes in it, hence the two famous types of ABE (i.e. Ciphertext-Policy ABE and Key-Policy ABE). In a general ABE scheme, we either have a single attribute authority or multiple attribute authorities where each of them is responsible of a certain set of attributes. The scheme is called *decentralised* if there is no reliance on a central authority.

[1] In a selective security model, for instance, selective-ID, the adversary has to choose the ID that he wants to attack ahead of the game.

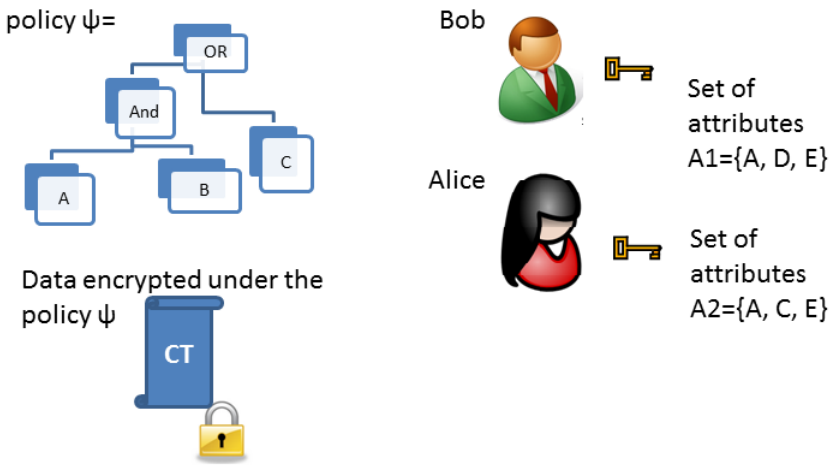


Figure 2-3: Ciphertext-Policy Attribute Based Encryption

More specifically, in Ciphertext-Policy ABE [BSW07], a user who has a set of attributes that satisfies the policy under which the data is encrypted, can decrypt the ciphertext. For instance, in Fig 2-3, as $\Psi(A_1) = 0$ and $\Psi(A_2) = 1$, Alice can then decrypt the ciphertext CT but Bob cannot. In the second type of ABE, Key-Policy ABE [ALdP11], the encryptor encrypts the data with respect to some attributes but the decryptor now has a key that corresponds to a certain policy. He can decrypt a certain ciphertext if the policy attached to his secret key is satisfied by the attributes embedded in the given ciphertext.

For instance, a very useful feature in CP-ABE is that encryption of some data with respect to some policies can happen even before the users have obtained their private keys. Thus, data can be encrypted without knowledge of the *actual* set of users who will be able to decrypt; it only specifies the policy which allows to decrypt. Therefore, future users can still decrypt some encrypted data as long as they have

enough attributes to satisfy the underlying policy.

2.5.3 Attribute Based Signatures (ABS)

In traditional digital signature schemes, the recipient of a signature is convinced that a particular signer has indeed authenticated the message in question. In Attribute-Based Signatures (ABS) [MPR08, MPR11], messages are signed with respect to a signing policy expressed as a predicate. Thus, the recipient is convinced that someone with a set of attributes satisfying the signing predicate has indeed authenticated the message without learning the identity of the signer or learning how the predicate was satisfied (i.e. which set of attributes was used in the signing). Furthermore, users cannot collude to pool their attributes together.

There are many applications of attribute-based signatures such as attribute-based messaging, e.g. [BFK$^+$06], trust negotiation, e.g. [FLA06], and leaking secrets. Refer to [MPR11] for more details and comparison with related notions such as mesh signatures [Boy07] and anonymous credentials [CL01].

Besides correctness, the security of attribute-based signatures requires signer privacy and unforgeability. Informally, signer privacy (sometimes is also referred to as anonymity), requires that a signature reveals neither the identity of the signer nor which set of attributes was used to satisfy the associated predicate. On the other hand, unforgeability requires that a signer cannot forge a signature with respect to a signing predicate that her individual attributes do not satisfy, even if she colludes with other signers.

Attribute Based Signature is a special signature scheme that can be considered as the signature version in the attribute based cryptography field. In ABS, an attribute authority or multiple ones, delegate their signing power to some signers based on the attributes which they have (e.g. Professor at Computer Science Department, Chancellor, etc.). A policy Ψ is a well defined relation (e.g. predicate) between these attributes. The signer can sign a certain message m with respect to a policy Ψ if he has enough attributes to satisfy it. Let $\mathcal{A}$ be a set of attributes for which a certain signer holds the corresponding secret keys, i.e $\mathsf{sk}_{\mathcal{A}}$, the signer can sign a message m with respect to Ψ using the secret key $\mathsf{sk}_{\mathcal{A}}$ iff $\Psi(\mathcal{A}) = 1$. Variants of attribute-based signatures exist in the literature each supporting policies that differ in their expressiveness. Those can be categorized into three main types of policies: non-monotonic

policies, e.g. [OT11], monotonic policies, e.g. [MPR11], and threshold-based policies, e.g. [LK08, SSN09, LAS$^+$10, HLLR12, GNSN12]. Schemes with more expressive policies are more interesting since they cover a larger scale of potential applications. Nevertheless, their current state–of–the–art instantiations are less efficient. The size of the signatures in existing instantiations of those supporting "monotonic" and "non-monotonic" policies, in the best case, are linearly dependent on the number of attributes in the policy [MPR11, OT11]. While the works of [HLLR12, GNSN12] yield constant-size signatures, they only support threshold policies.

Early proposals of attribute-based signatures considered the case of multiple attribute authorities where each authority is responsible for a sub–universe of attributes [MPR08, OT11]. However, the multi–authority case still had the problem of relying on the existence of a central trusted authority. Moreover, in some cases, the security (unforgeability) of the whole system is compromised if the central authority is corrupted. Okamoto and Takashima [OT13] recently proposed the first decentralized construction.

Traceability in attribute-based signatures was first addressed by Khader [Kha07] who proposed the notion of attribute-based group signatures. In this notion, only the anonymity of the identity of the signer is preserved, whereas the attributes used are not hidden. This is an undesirable property for many applications. Later, Khader et al. [KCD09] proposed a traceable attribute-based signature scheme that relies on the verifier to decide the policy and thus requiring interaction in the signing protocol. Even though this can be useful in certain applications (see [KCD09] for details), such interaction is prohibitive for many applications. A more recent construction by Escala et al. [EHM11] adds the traceability feature (it was called revocation by the authors) to standard ABS schemes. The proposed scheme in [EHM11] is in the inefficient composite-order groups setting and was originally proven in the Random Oracle Model (ROM) [BR93]. Although the authors described how the reliance on random oracles could be removed, this was done informally and without a concrete construction or a full security proof. In addition, their construction relies on a central attribute authority which could be a bottleneck when the number of members of the system increases.

Below, we present a table of comparison of existing ABS schemes in terms of dealing with the following features; generic construction, full anonymity (i.e. to hide the identity and the signing attribute of the signers), traceability, decentralisation, user-controlled linkability, and hidden policy.

Table 2.1: Existing ABS schemes and their features

Feature	[Kha07]	[OT13, OT11]	[MPR08]	[EHM11]
Generic Construction	✘	✘	✓	✘
Full Anonymity	✘	✓	✓	✓
Traceability	✓	✘	✘	✗ [1]
Decentralisation	✘	✓	✘	✘
UCL	✘	✘	✘	✘
Hidden Policy	✘	✘	✘	✘

2.6 Conclusion

In this chapter, we gave a cryptographic background that is needed in the rest of the thesis. We first presented symmetric/asymmetric encryption notions and their related security definitions (CPA, CMA, CCA, and AE). We also discussed different types of digital signature schemes, e.g. group signatures, ring signatures. Moreover, we gave the definition of *provable security*, showed how it is based on hardness assumptions, and discussed the difference between the two types of security models, namely, *random oracle model* and *standard model*. At the end, we gave an introduction to the main topic that we are going to deal with in this thesis, i.e. Attribute based signatures.

[1] It actually offers a weak version of traceability, we show in 4.2.2 how to violate its security model.

Chapter 3

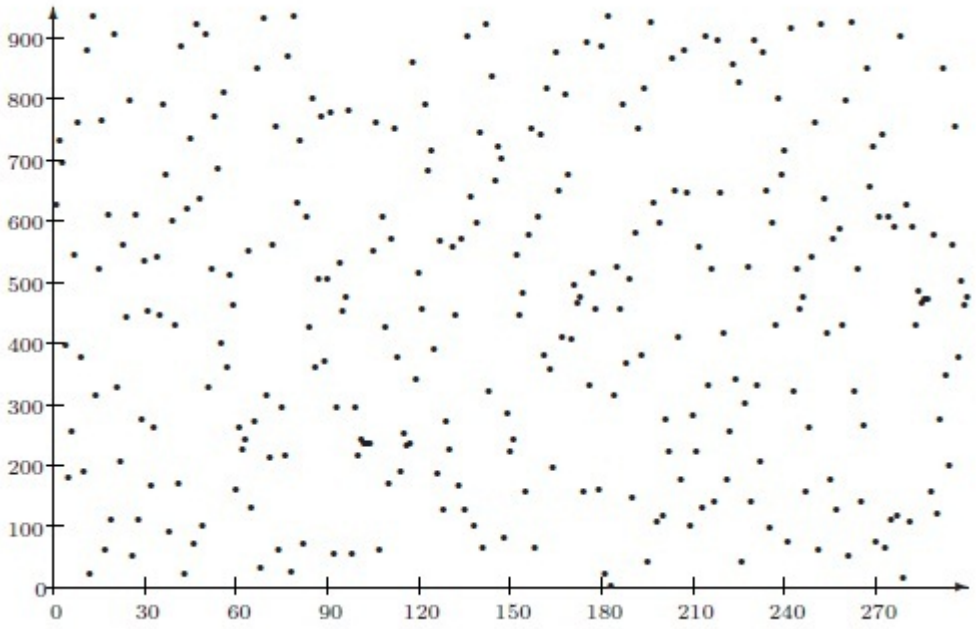
Mathematical Background

3.1 Introduction

In this chapter, we define the Discrete Logarithm Problem (Dlog) and some of its variants. The *Dlog family of assumptions* has been heavily used in cryptography along with factorization and lattices. We show how the difference between the levels of difficulty of Dlog relies on the algebraic groups in which we try to solve it. Moreover, we give some recent results on solving the Dlog in some fields. We have also created a useful reduction diagram to give an idea about the relationships between Dlog and some of its variants. In this chapter, we focus on groups of points on a *suitable* elliptic curve. For this, we first define elliptic curves, Bilinear Pairings, and then move on to show the properties of special elliptic curves that preserve a good level of difficulty of Dlog and at the same time are considered to be pairing-friendly, since pairings have become very popular and used in most of the cryptographic schemes. We then generalize bilinear maps to define multilinear maps, yet another useful cryptographic module that has been recently realised after a decade of being a part of the open problems' world.

The last part of this chapter will be about the Non-Interactive Zero Knowledge proofs (NIZK proofs), in which a prover can prove to a verifier that he surely knows a witness for an NP-problem but without letting the verifier gain any extra information beyond this fact, hence the zero knowledgeness. Note that NIZK proofs are widely used in recent cryptographic schemes. Finally, we present different existing types of NIZKs.

Figure 3-1: Exponential Function Behaviour



3.2 Discrete Logarithm Problem Dlog

DLog in an abstract group $(\mathbb{G}, \star)$: Let $\mathbb{G}$ be a group whose group law we denote by the symbol $\star$. The Discrete Logarithm Problem for $\mathbb{G}$ is to determine, for any two given elements g and h in $\mathbb{G}$, an integer a satisfying;

$$\underbrace{g \star g \star g \star \cdots \star g}_{(a\ times)} = h$$

Dlog in a multiplicative group $(\mathbb{G}, \times)$: Let $\mathbb{G}$ be a finite cyclic multiplicative group with n elements. Let g be a generator of $\mathbb{G}$; then every element $h \in \mathbb{G}$ can be written in the form $h = g^x$ for some integer $x \in \mathbb{Z}_n$. Thus,we can define the Discrete Logarithm function for the generator g as:

$\text{Dlog}_g : \mathbb{G} \to \mathbb{Z}_n$, where $\text{Dlog}_g(h) = x \in \mathbb{Z}_n$

To have an idea about the behaviour of the Dlog, let's have a look at the scatter graph in Figure 3-1 [PHP08]; in this example, we have the set of points representing $f(x) = 627^x \bmod 941$ for $x = 1, 2, 3, \ldots$.

Hardness of Dlog in different groups Let's state some proved facts about the hardness of the DLog in different groups.

- In $(\mathbb{Z}_n, +)$, the discrete logarithm problem Dlog can be solved in linear time.
- In $\mathbb{F}_{p^n}^*$ the discrete logarithm problem can be solved in sub-exponential time. We can distinguish between two Number Field Sieve algorithms, one in large characteristic fields [Sch99], which takes time

  $$L_{p^n}(1/3, (64/9)^{1/3})$$

  and the second in low characteristic fields [JL02], which takes time

  $$L_{p^n}(1/3, (32/9)^{1/3})$$

  where;

  $$L_{p^n}(\beta, c) = exp\left((c + o(1))(\log p^n)^{\beta}(\log(\log p^n)^{1-\beta}\right)$$

  Note that, as per [PSV06], no data points are available for computing discrete logarithm in $\mathbb{F}_{p^6}$ where p is a large prime.
- In a subgroup of an elliptic curve[1] over a finite field $\mathbb{K}$, $E(\mathbb{K})$, by Pollards rho method [Pol78], takes time $O(\sqrt{|\mathbb{K}|})$, which is based on the birthday paradox.

So, from a quick look at the above results, one can tell how important it is to make the right choice for a group in order to make the Dlog the hardest possible. Roughly speaking, one cannot ignore the other factors that play a tremendous role in choosing such a group, like how hard it is to do group operations, inversion, and hashing into the group, but the most important part lies in the achieved security level. In a given cryptosystem, we have the length of the keys, and we also have the level of security of the cryptosystem based on the fastest known computational attack against such a cryptosystem. We usually compare the security of public key cryptosystems to symmetric key cryptosystems. We can state some facts to help clarify this idea, and why elliptic curve cryptosystems ECC can be considered as a better choice when it comes

[1] Note that some elliptic curves have shown serious vulnerabilities against Dlog, like supersingular elliptic curves with small embedding degree [MOV93] or where $|E(\mathbb{F}_p)| = p$ [Sma99]. See 3.4.3 for more details.

to having shorter keys. In [BKK+09] they give an explanation of the relation between length of the keys and the security level, saying that 160-bit key for an ECC yields an 80-bit security whereas RSA takes 1024 bit length moduli in order to give same security level[1]. In other words, for symmetric key algorithms, we have that *block ciphers* with n–bit key are secure against $\approx 2^n$–time attacks. We also know that *hash functions* with n-bit output are secure against $\approx 2^{n/2}$–time attacks. It is very important to figure out what is the current key length recommendation with respect to Dlog and Factoring based cryptosystems. Does *factoring* a modulus of length n take 2^n time? Does computing Dlog in a group with 2^n elements take 2^n time? Based on the previous example that we have just mentioned, it is clearly not the case. The keys will be much longer when considering asymmetric key algorithms. To make things clear, we will have yet another example, achieving 112 bit security.

Example 1. . *According to NIST, to achieve* 112*–bit security, we should have the following:*

- *For factoring, we need a* 2048*-bit modulus.*
- *For Dlog in order-*q *subgroup of* $\mathbb{F}_p^*$*, we need* $\|q\| = 224$*,* $\|p\| = 2048$*.*
- *For Dlog in an elliptic curve group of order* q*, we need* $\|q\| = 224$*.*

3.2.1 Recent results on Dlog

To give the reader an idea about the current situation of the Dlog hardness, we list some of the recent results that show the state-of-the-art of Dlog in different cases:

- In [BBD+13], authors use the function field sieve algorithm to solve the *Dlog* in GF(2^{809}), which is a prime field degree.
- In [Jou13b], Joux was able to compute discrete logarithms in GF(2^{6168}) = GF(2^{257*24}) using less than 550 CPU hours.
- Dlog in Medium Prime Case: In [Jou13a], they introduce a new technique called *Pinpointing*, which allows them to construct multiplicative relations much faster,

[1] See Algorithms, key sizes and parameters, Enisa 2014, for more details on current recommendations for key size in different cryptosystems [Eni14]. Another useful website for recommended key sizes is [Gir14]

thus reducing the asymptotic complexity of relations' construction. They show the *feasibility* of their method with discrete logarithm records in two medium prime finite fields, the first of size 1175 bits and the second of size 1425 bit (GF(p^{47}), GF(p^{57}), p is a 25-bit prime)

How does this affect current cryptosystems that are in use?

In pairing-based cryptography (section 3.5), if one wants to use Type-1 pairing, by using supersingular curves (of genus 1 or 2), for instance, in characteristic 2, one would have a group over GF(2^p) for some prime $(e.g., p = 257)$, now the pairing would map into an extension of this field of small degree $k \leq 6$, which is called the MOV attack [MOV93] (see paragraph 3.4.3). Hence, the target group would be contained in GF(2^{p*k}).

3.3 Diffie–Hellman Problem

Many other problems that are related to DLog are believed to be hard as well. We will first define the original Diffie–Hellman problem, which is directly related to DLog, then we discuss different versions of Diffie–Hellman. It was the insightful idea of Diffie–Hellman key exchange algorithm that gave a brilliant new direction in cryptography [DH76], called public key cryptography, where there is no need any more to exchange the key secretly. Now Bob and Alice can exchange a common secret session key over insecure channel, where the adversary Eve can see both their public keys, but doesn't have enough computational power to get their "secret key" efficiently (e.g. in polynomial time).

What is the Diffie–Hellman problem? Alice and Bob can easily share the public parameters ($\mathbb{G}, g$, etc.), where g is a generator of a group $\mathbb{G}$ (e.g. group of points on a well chosen elliptic curve). Bob thinks of a value x, computes g^x and sends it to Alice. Alice in her turn, does the same by choosing a value y, computing g^y and sending it back to Bob. Both of them can easily work out g^{xy} and have it as their shared secret session key. Now Eve can see g, g^x and g^y. But the question that Diffie and Hellman asked in their work is, can Eve efficiently compute g^{xy}? This problem is what is known as the Computational Diffie–Hellam problem (CDH). There is also the decisional version of it, below are their formal definitions:

CDH The Computational Diffie–Hellman problem CDH is as follows:
Given $g, g^a, g^b \in \mathbb{G}$, where $a, b \in \mathbb{Z}_p$.
Compute g^{ab}.

DDH The Decisional Diffie–Hellman problem DDH is as follows:
Given $g, g^a, g^b, g^c \in \mathbb{G}$, where $a, b, c \in \mathbb{Z}_p$.
Decide if $g^c = g^{ab}$.

3.3.1 Decisional, Gap, Strong, Hashed, and Twin Notions

We will define here weaker variants of the Diffie–Hellman problem, or what can also be called stronger assumptions than the original computational and decisional DH assumptions. It seems to be a good idea to generally define some useful notations which are going to be integrated into the original DH problem to make new useful assumptions.

– Decisional For every computational problem there exists a decisional version of it [Bon98], which is, instead of computing the solution of a problem instance, the problem will be to check whether a given element of a specific group is a solution of this problem without necessarily solving the problem.

– Gap This is the idea of separating the Decisional problems from computational ones in cryptographic groups [JN03]. That happened after they found that decisional problems can be polynomially solved in certain groups. So they wanted to make sure that the computational problem stays hard in the same groups where the decisional problem is easy. Here comes the idea of the Gap, which measures the hardness of the computational problem for an adversary that has already a full access to a decisional oracle that solves the decisional version of the problem.

– Hash The hash can be integrated into decisional problems [GKR04], let's say the DDH, so instead of distinguishing between a random element of a specific group and a solution of a problem instance, the problem will be to to distinguish between a random string and a hashed solution of the problem.

– Twin The Twin concept for any problem instance is to compute it twice based on a certain given [CKS08]. So instead of having two variables and the question is to compute their product, the given will be three variables, so the question will be to to find the products of two chosen pairs from what is given.

In the coming sections, we will further explain elliptic curves and pairings, and later come back to different existing hardness assumptions where some of them involve pairings; we will also give a diagram of the reduction relationships between them.

3.4 Elliptic Curves

We first define *projective planes* then we define *elliptic curves*.

Definition 2. *[Sma13]: Let $\mathbb{K}$ be any field. Consider the set of triples (X, Y, Z), where $X, Y, Z \in \mathbb{K}$ are not all simultaneously zero. Such a triple is called a projective point. On these triples, define an equivalence relation $(X, Y, Z) \equiv (X', Y', Z')$ if there exists a $\lambda \in \mathbb{K}$ such that $X = \lambda X', Y = \lambda Y'$ and $Z = \lambda Z'$. The projective plane $\mathbb{P}^2(\mathbb{K})$ over $\mathbb{K}$ is defined as the set of equivalent classes (X, Y, Z).*

Definition 3. *[Gal12] An elliptic curve over a field $\mathbb{K}$ is the set of solutions in the projective plane $\mathbb{P}^2(\mathbb{K})$ of a non-singular (see next section) Weierstrass equation*

$$E : y^2z + a_1xyz + a_3yz^2 = x^3 + a_2x^2z + a_4xz^2 + a_6z^3$$

where $a_i \in \mathbb{K}, i = 1, \cdots, 6$.

The set of $\mathbb{K}$-rational points on E, i.e. the solutions in $\mathbb{P}^2(\mathbb{K})$ to the above equation, is denoted by $E(\mathbb{K})$. Notice that the curve has exactly one rational point with coordinate Z equal to zero, namely $(0, 1, 0)$. This is the point at infinity, which will be denoted by $\mathcal{O}_E$. The affine version of the Weierstrass equation is:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

The $\mathbb{K}$-rational points in the affine case are the solutions to E in $\mathbb{K}^2$, plus the point at infinity $\mathcal{O}_E$.

Let us see the difference between singular and non-singular Weierstrass equations, and how they change according to the characteristic of the underlying field $\mathbb{K}$.

3.4.1 Singular vs non-Singular (smooth) Weierstrass equation

In general, for any equation, the singular points are those at which all the partial derivatives simultaneously vanish [Gal12]. We say that the *discriminant* of the equation should be different from zero. The discriminant of a polynomial is the product of the squares of the differences of the polynomial roots. In particular, for a Weierstrass equation, its singularity depends on the characteristic of the underlying field on which it is defined. For example, if $E(x, y)$ is defined over $\mathbb{K}$, then we can distinguish between the following cases: (See [Gal12] for proofs)

- If char($\mathbb{K}$)$\neq 2, 3$, then every Weierstrass equation over $\mathbb{K}$ is isomorphic over $\mathbb{K}$ to a Weierstrass equation

  $$y^2z = x^3 + a_4xz^2 + a_6z^3$$

  for some $a_4, a_6 \in \mathbb{K}$. It is called the short Weierstrass form, and this equation is non-singular if and only if the discriminant $-16(4a_4^3 + 27a_6^2) \neq 0$ in $\mathbb{K}$.

- If char($\mathbb{K}$) =3, then every Weierstrass equation over $\mathbb{K}$ is isomorphic over $\mathbb{K}$ to a Weierstrass equation

  $$y^2z = x^3 + a_2x^2z + a_4xz^2 + a_6z^3$$

  is non-singular if $a_2^2(a_4^2 - 4a_2a_6) - a_4^3 \neq 0$

- If char($\mathbb{K}$) = 2, then every Weierstrass equation over $\mathbb{K}$ is isomorphic over $\mathbb{K}$ to a Weierstrass equation

  $$y^2z + xyz = x^3 + a_2x^2z + a_6z^3$$

  which is non-singular if $a_6 \neq 0$ or to

  $$y^2z + yz^2 = x^3 + a_4xz^2 + a_6z^3$$

  which is non-singular for all $a_4, a_6 \in \mathbb{K}$.

Definition 4. *The Group Law (+) on an Elliptic Curve [Sma13]: Let E denote an*

elliptic curve given by

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x_2 + a_4x + a_6$$

and let $P_1 = (x_1, y_1)$ *and* $P_2 = (x_2, y_2)$ *denote points on the curve. Then*

$$-P_1 = (x_1, -y_1 - a_1x_1 - a_3).$$

Set

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1},$$
$$\mu = \frac{y_1x_2 - y_2x_1}{x_2 - x_1}$$

when $x_1 \neq x_2$*, and set*

$$\lambda = \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3},$$

$$\mu = \frac{-x_1^3 + a_4x_1 + 2a_6 - a_3y_1}{2y_1 + a_1x_1 + a_3}$$

when $x_1 = x_2$ *and* $P_2 \neq -P_1$*. If*

$$P_3 = (x_3, y_3) = P_1 + P_2 \neq \mathcal{O}$$

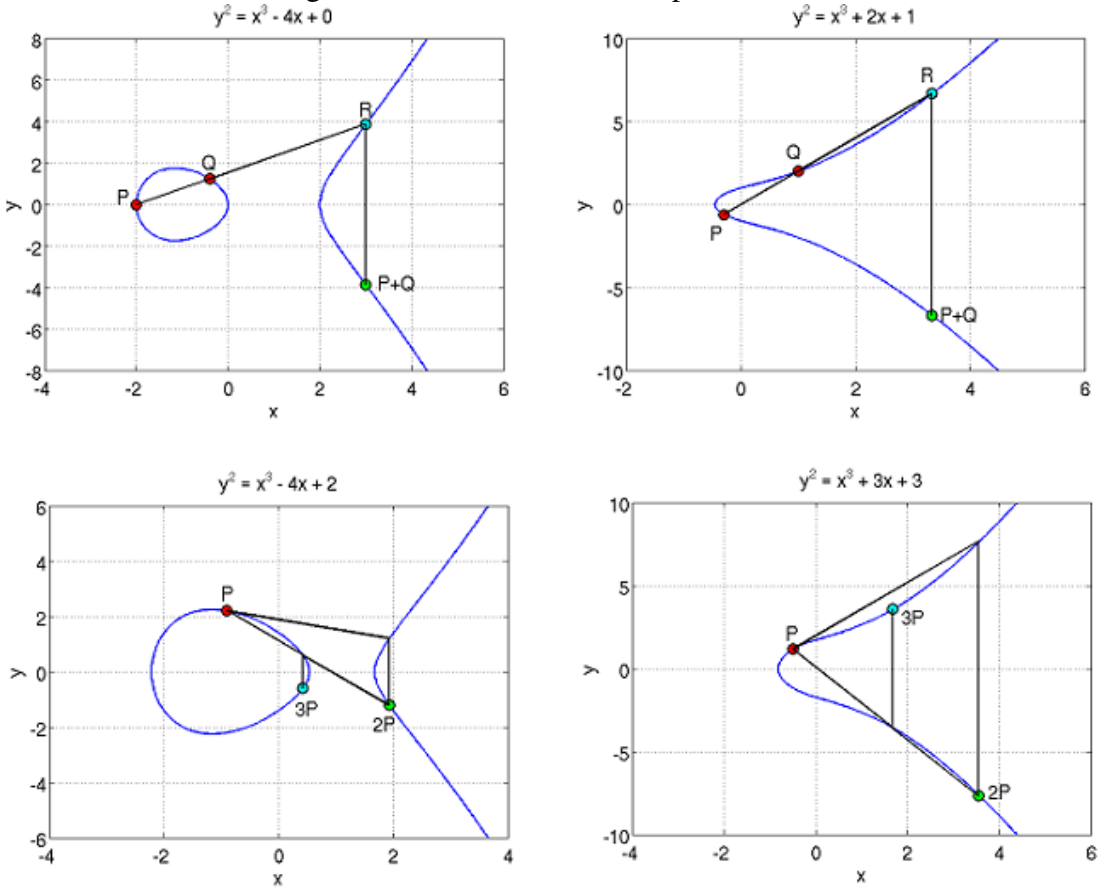
then x_3 *and* y_3 *are given by the formulae*

$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2,$$
$$y_3 = -(\lambda + a_1)x_3 - \mu - a_3.$$

Definition 5. *Group of points on an Elliptic Curve [Gal12]: The group of points on an elliptic curve consists of the set of affine points of the curve plus the point at infinity* $\mathcal{O}_E$ *which represents the identity element of the group. The law of the group is the aforementioned addition.*

Example 2. *Addition on an elliptic curve: Figure 3-2 shows how the addition of two points on an elliptic curve can be geometrically illustrated.*

Figure 3-2: Addition on an elliptic curve



3.4.2 Ordinary vs Supersingular Elliptic Curves

To begin with, both *supersingular* and *ordinary* curves are non singular curves [Gal12]. Now, for any prime power q, we let $\mathbb{F}_q$ denote the field of q elements. We give a couple of useful definitions/theorems that we are going to use later on in this section.

Definition 6. *(q-Frobenius map)[Gal12] Let p be a prime and let $q = p^m$ for some $m \in \mathbb{N}$. Let E be an elliptic curve over $\mathbb{F}_q$. The q-Frobenius map is the rational map $\pi_q : E \to E$ such that $\pi_q(\mathcal{O}_E) = \mathcal{O}_E$ and $\pi_q(x, y) = (x^q, y^q)$.*

Theorem 1. *(Hasse bound) [Gal12] Let E be an elliptic curve over $\mathbb{F}_q$ and denote t the trace of q-Frobenius map. Then $|t| \leq 2\sqrt{q}$.*

Theorem 2. *[Gal12] The number of points on an elliptic curve over $\mathbb{F}_q$ lies in the Hasse interval $[q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$, therefore $t = q + 1 - \#E(\mathbb{F}_q)$.*

Definition 7. *(Embedding degree)[Gal12] Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$ and let n be a prime dividing $\#E(\mathbb{F}_q)$. The embedding degree of E with respect to n is the smallest integer k such that n divides $q^k - 1$.*

Definition 8. *[Gal12] An elliptic curve E defined over a field $\mathbb{F}_q$ of characteristic p is supersingular if $p|t$, where $t = q + 1 - \#E(\mathbb{F}_q)$. If $p \nmid t$ then E is ordinary.*

Supersingular curves have small embedding degree k, $k = 2$ for prime fields, and $k \leq 6$ for others. They also support a distortion map $\Phi : E(\mathbb{F}_q) \rightarrow E(\mathbb{F}_{q^k})$. A comparison of a family of ordinary curves (MNT) [MNT01] and supersingular curves is given in [PSV06], in which they consider two examples of cryptosystems, namely, the (BLS) short signature scheme of Boneh, Lynn and Shacham [BLS04b] and the identity based encryption scheme (IBE) of Boneh and Franklin [BF01].

3.4.3 Attacks on Vulnerable Curves

Anomalous attack When an elliptic curve $E(\mathbb{F}_{p^m})$ has a trace of Frobenius equal to 1, the number of points on the curve equals p^m. In this case the elliptic curve is said to be $anomalous$ and there exists an efficient reduction which enables an attacker to solve the discrete logarithm problem on E in polynomial time [Sma99]. This method works by transporting the discrete logarithm problem on E to the discrete logarithm problem on the additive group of $\mathbb{F}_p$, where the discrete logarithm problem can be solved in linear time.

GHS attack This attack works for some elliptic curves defined over a finite field of characteristic two. It was shown by Gaudry, Hess and Smart [GHS02] that the elliptic curve discrete logarithm problem ECDLP on an elliptic curve $E(\mathbb{F}_{2^n})$ can be reduced to the discrete logarithm problem on an associated hyperelliptic curve[1] $C(\mathbb{F}_{2^m})$ where $m < n$ (i.e. the associated hyperelliptic curve is defined over a smaller field). This is done through a process known as the Weil descent. In some instances, where the genus of C is large enough, there exists a sub-exponential algorithm to solve the discrete logarithm in C.

[1]Hyperelliptic curves are a special class of algebraic curves and can be viewed as generalizations of elliptic curves, see [MZW96] for more details.

MOV-reduction and FR-reduction In [MNT01], the authors give a good comparison of both well known attacks FR-reduction [FR94] and the MOV attack [MOV93] that actually embed a subgroup $\mathbb{G} \subset E(\mathbb{F}_q)$ to $\mathbb{F}^*_{q^k}$ for an extension field $\mathbb{F}_{q^k}$ and reduce ECDLP based on $\mathbb{G}$ to Dlog based on a subgroup of $\mathbb{F}^*_{q^k}$. Let n =order($\mathbb{G}$), and the *n-torsion subgroup* is denoted as $E[n] = \{P \in E | nP = \mathcal{O}\}$.

- In MOV-reduction, ECDLP on $\mathbb{G}$ is reduced to Dlog for the smallest integer k such that $E[n] \subset E(\mathbb{F}_{q^k})$. As a result, ECDLP in groups defined over supersingular curves can be efficiently reduced to Dlog in $\mathbb{F}^*_{q^k}$ for $k \leq 6$.
- In FR-reduction, ECDLP on $\mathbb{G}$ is reduced to DLP for the smallest integer k such that $n | q^k - 1$.

It is easy to see that in general, MOV $\Longrightarrow$ FR-reduction. But when n is prime and $n \not| q - 1$ then, $E[n] \subset E(\mathbb{F}_{q^k}) \Longleftrightarrow n | q^k - 1$, and therefore MOV $\Longleftrightarrow$ FR-reduction.

Recent MOV attack on Supersingular curves over field of characteristic 3 In [SSHT12], the authors estimate the time complexity of solving Dlog over $GF(3^{6n})$ for $n = 193, 239, 501$ as $2^{72}, 2^{78}, 2^{111}$ respectively. This estimation is based on their result in which they solved the Dlog over $GF(3^{6 \cdot 97})$ in 148.2 days by using 252 CPU cores using η_T pairing [HSV06]. The method which they use is an improved version of the function field sieve (FFS), which was first proposed in [Adl94], 1994. In [SSHT12], they describe their computational result as a contribution to the secure use of pairing-based cryptosystems with the η_T pairing by saying that n should be greater than 239 to keep 80 bit security.

3.5 Bilinear Maps or Pairings

The first time pairings were used in Cryptography was in 1993, when Menezes *et al.* [MOV93] proposed their famous MOV-reduction to reduce the ECDLP to the Dlog. Thereafter, in 1994 and 1999, Frey *et al.* [FR94] proposed the FR-reduction to also reduce the ECDLP to the Dlog. Hence, the use of pairing was first considered to be *destructive* in the sense that it can only be used to break elliptic curve cryptosystems. The first time when pairings were used for "good" was in 2000 due to Joux [Jou00][1].

[1]Well, same thing happened with lattice-based cryptography, it was first used for "bad", i.e. in cryptanalysis, the LLL algorithm [LL82] to factor polynomials, whereas in 1986, people started using

Shortly after Joux's work, the pioneering work of Boneh-Franklin [BF03] appeared in 2003, which was actually behind this revolution of pairing–based cryptosystems. In [BF03], they built the first practical identity based encryption after many years of Shamir's thought on IBE in 1984 [Sha84]. Pairings that were used in the original BF scheme was renamed later, by Chatterjee-Menezes [CM09] as *Type-1* pairings, that can only be used with supersingular elliptic curves. Later on, some vulnerabilities were found with supersingular curves. As a result, people became more interested in studying ordinary curves, how to construct them and how to define pairing-friendly curves. The conditions are not easy to satisfy, for instance, a curve should have a relatively high embedding degree to resist against MOV-attack, but at the same time stay computationally useful. Note that efficient bilinear maps can also be built over groups of points on hyperelliptic curves, the general case of elliptic curves, however, we will focus here on the simple and practical type of pairings which use elliptic curves.

3.5.1 Types of Pairings

A *bilinear map* can be defined as a function that maps any pair of elements from two given groups (e.g. groups of points on an elliptic curve) to an element in another group (subgroup of a multiplicative group of a finite field, which is the case for the Tate Pairing).

Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be three groups of the same prime order p, a *pairing* is an efficiently computable function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, satisfying that:

1. $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$, for all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ and all $a, b \in \mathbb{Z}_p$
2. *Non-degeneracy*, which is, if g_1 is a generator of $\mathbb{G}_1$, g_2 is a generator of $\mathbb{G}_2$ then $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$.

According to the existence of an isomorphism ψ: $\mathbb{G}_2 \to \mathbb{G}_1$ and ψ^{-1} we can define three general types of bilinear mappings as by [CM09].

- Type-1: if ψ and ψ^{-1} both are efficiently computable, then we can look at $\mathbb{G}_1$ and $\mathbb{G}_2$ as the same group, i.e. $\mathbb{G}_1 \approx \mathbb{G}_2$ effectively. We can define the pairing as $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Type-1 is called *symmetric* bilinear pairings. When $\mathbb{G}_1 \neq \mathbb{G}_2$ we can distinguish between two types of "asymmetric" pairings that are Type-2 and Type-3.

Lattices in a constructive way, i.e. to build cryptosystems.

- Type-2: Here only ψ is efficiently computable. A typical example is to take $\mathbb{G}_1 \subset E(\mathbb{F}_q)$ and $\mathbb{G}_2 \subset E(\mathbb{F}_{q^k})$, the extension field of $E(\mathbb{F}_q)$, therefore the *Trace* map will play the role of the isomorphism ψ [1].

- Type-3: if no efficiently computable ψ or ψ^{-1} are known, then we call it Type-3. As by [GPS08], this type has some advantages over Type-2 in terms of the elements size, group operations, and hashing into $\mathbb{G}_2$.

3.5.2 Pairing-Friendly Elliptic Curves

A great reference on pairing-friendly elliptic curves is Freeman *et al.*'s work [FST10], in which they formalize and classify previous work already done in this area by Miyaji *et al.* [MNT01], Barreto *et al.* [BLS03], Scott *et al.* [SB06], and many other important relevant work. As we mentioned earlier, the early cryptographic work that used pairings focused on pairings of Type-1, which seemed to be only feasible with supersingular elliptic curves. With the presence of MOV and FR reductions (see 3.4.3), along with the recent records on solving Dlog in multiplicative fields of small characteristic (see 3.2.1), people had to think of using ordinary elliptic curves to build pairings. Note that in these reductions, curves with small embedding degree k are more vulnerable. Also note that the best known attack on ECDLP is the parallelized Pollard rho algorithm [Pol78] which has running time $\mathcal{O}(\sqrt{r})$ where r is the size of the largest prime-order subgroup of $E(\mathbb{F}_q)$, whereas the best known attack on Dlog in finite fields is the index calculus which has running time subexponential in the field size [SWD96]. As they show in [FST10], in order to achieve the same level of security in both groups, the size q^k of the extension field must be significantly larger than r. Let ρ be the parameter that measures the base field size relative to the size of the prime order subgroup , i.e. $\rho = \log(q)/\log(r)$. Note that $\log(q^k)/\log(r) = k \cdot \rho$. Now, let's reconsider the 112–bit security level given in example 1, if one wants to achieve 112–bit security level, we have to set up a system with a subgroup of a size $r = 224$, and a $2200 \leq q^k \leq 3600$–bit extension field. Therefore, one can choose between ($\rho \approx 1$ and $10 \leq k \leq 16$) or ($\rho \approx 2$ and $5 \leq k \leq 8$). Note that the embedding degree can't be a large number if we want the pairings to be practical. On the other hand, small ρ values

[1] This type is widely used in cryptographic protocols although more research has been made [CM09] and [SV07] to study the real necessity of the existence of ψ and the benefits that it offers in terms of functionality, security, performance and whether or not Type-3 can always be used instead of Type-2.

are in fact desirable in order to speed up the arithmetic on the elliptic curve. Of course, the ideal situation is when $\rho = 1$. Informally speaking, a pairing-friendly elliptic curve is an elliptic curve with a small embedding degree and a large prime-order subgroup. We will distinguish between the two cases, i.e. supersingular and ordinary curves, as follows:

Supersingular curves have been classified in three types:

- If $q = p$ a prime, maximum $k = 2$
- If $q = 2^m$, maximum $k = 4$
- If $q = 3^m$, maximum $k = 6$

In the last two, $\mathbb{F}_q$ must be larger due to Coppersmith's index calculus method for Dlog in finite fields of small characteristic [Cop84]. However, Kolbitz-Menezes in [KM05] support the idea that ordinary curves with small embedding degree are not necessarily more secure than the supersingular curves with the same embedding degree against the MOV [MOV93] and FR [FR94] attacks. With the first choice, although it is not affected by Coppersmith's index calculus method [Cop84], the embedding degree is 2, which is very small, therefore p should be really large to achieve certain security level, and this makes working in the base field almost impractical. In Freeman *et al.*'s paper [FST10], they classify pairing-friendly curves, according to their construction, to families and non-families and formally defined the term "pairing-friendly".

Definition: suppose E is an elliptic curve defined over finite field $\mathbb{F}_q$. We say that E is *pairing-friendly* if the following two conditions hold:

1. There is a prime $r \geq \sqrt{q}$ dividing $|E(\mathbb{F}_q)|$,
2. The embedding degree of E with respect to r is less than $\log_2(r)/8$.

That is why supersingular curves are always considered as pairing-friendly curves if they have large prime-order subgroup since the embedding degree k is always less than or equal to 6. The fact that supersingular curves with embedding degree 6 are defined over fields of small characteristic made the achievement of a higher level of security necessarily through constructing pairing-friendly ordinary curves. On the other hand, ordinary curves are defined as families, of two types, sparse and complete (definitions below)[1]. As per [FST10], a pairing-friendly ordinary curve can be constructed if and only if all the following conditions hold:

[1] According to [FST10], there is no such thing as family of supersingular curves.

1. q is prime or prime power.
2. r is prime.
3. The trace of the curve t is relatively prime to q (i.e. t and q are co-prime).
4. r divides $q + 1 - t$.
5. r divides $q^k - 1$, and r doesn't divide $q^i - 1$ for $1 \leq i < k$.[1]
6. $4q - t^2 = Dy^2$ for some sufficiently small positive integer D and some integer y.

Where D is the the CM discriminant of the curve, it should be small in order to be able to find an equation of such a curve.

So the idea of finding or constructing a curve can be summarized by two steps; firstly, we fix k and compute some integers, t, r, q such that there is an elliptic curve $E(\mathbb{F}_q)$ that has an embedding degree k, trace t, and a subgroup of prime order r. The second step is actually performed by using the complex multiplication method [FST10] to find the equation of the curve defined over $\mathbb{F}_q$. Now if we want to generalize the construction method of a family of curves instead of constructing a single curve, one should write r, t, and q as polynomials $r(x)$, $t(x)$, and $q(x)$. This method has been used to construct MNT curves [MNT01], BLS curves [BLS03], and others. Remember that we need $q(x)$ and $r(x)$ to be primes. So we need to find x_0 for which $q(x_0)$ and $r(x_0)$ are both primes. But the only things that we know about prime values of polynomials are conjectures. That's what actually stops the process of constructing a family of paring-friendly ordinary curves from being an easy task. To differentiate between the *complete* families and the *sparse* ones, we can look at condition (6) of constructing families of pairing-friendly ordinary curves, $4q(x) - t(x)^2 = Dy^2$. If we can find a polynomial $y(x) \in \mathbb{Q}[x]$ that satisfies the equation, i.e. we can write y in terms of x, we say that the family is *complete*, otherwise, it is said to be *sparse*. For the condition (1), it becomes $q(x) = p(x)^d, d \geq 1$, and $p(x)$ represents primes (In the literature, most of the well known families of pairing-friendly curves have $d = 1$). The ρ value of the family will be defined as: $\rho = \lim_{x\to\infty} \frac{q(x)}{r(x)}$.

[1] Condition (5), which is in fact the definition of the embedding degree, can be replaced by [r divides $\phi_k(t-1)$].

For the *sparse* families, the well known families in the literature are the MNT [MNT01], GMV [GMV07], and Freeman [Fre06].

In the *complete* families we have three different families which are the Cyclotomic families, Sporadic families, and Scott-Barreto families as per [FST10]

According to Aranha *et al.* [AFCK$^+$12], the state-of-the-art of pairing friendly elliptic curves is as follows:

1. There are the MNT curves [MNT01], with $k = 3, 4, 6$, $\rho \approx 1$.
2. There are Freeman curves [Fre06] with $k = 10$ $\rho \approx 1$.
3. KSS [KSS08] curves: $k = 18$, $\rho \approx 4/3$
4. BN [BN06] curves: $k = 12$, $\rho \approx 1$
5. BLS12 [BLS03] curves: $k = 12$, $\rho \approx 1.5$
6. BLS24 [BLS03] curves: $k = 24$, $\rho \approx 1.25$

Note that all of the above are families of curves. In [AFCK$^+$12] they implemented asymmetric pairings of Type–3 derived from (KSS), (BN) and (BLS) elliptic curves at the 192-bit security level and found that (BLS12) is the fastest among the others. In the same paper, they showed that the pairings computation at the 192-bit security level is not as expensive as people previously thought.

To achieve the 192-bit security level, they gave the following requirements:

- In order to resist the Pollard's rho attach [Pol78] on ECDLP in $\mathbb{G}_1$, the bitlength of r should be at least 384.
- In order to resist the number field sieve attack [Adl94] on Dlog in $\mathbb{F}^*_{q^k}$, the bitlength of q^k should be at least 7680.

3.5.3 Pairings: Supersingular Curves Vs Ordinary Curves

We have previously defined the pairings in an abstract way. Here we will take the example of the Tate pairings and its modified version and see what the differences are when it's applied over the two types of curves, supersingular and ordinary [PSV06]. First let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ represent three finite abelian groups such that the Dlog is hard

in the three of them. As we mentioned earlier, we mean by a *pairing* a non-degenerate bilinear map:

$$t : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

Let $\mathbb{G}_1$ be a subgroup of $E(\mathbb{F}_q)$ of a prime order r, $\mathbb{G}_2 \subset E(\mathbb{F}_{q^k})$, k be the embedding degree, i.e. r divides $q^k - 1$, and $r \nmid q^s - 1, 0 < s < k$. If $P \in \mathbb{G}_1$, let f_P denote the function with divisor[1]

$$(f_P) = r(P) - r(\mathcal{O})$$

Now, we can define the *unmodified Tate pairing* t as following, if $Q \in \mathbb{F}_{q^k}$ then;

$$t(P, Q) = f_P(D_Q),$$

where D_Q is a divisor equivalent to $(Q) - (\mathcal{O})$.

The original method to calculate the pairing is due to Miller [Mil86][2], while lots of optimizations that have been applied to it later in both types of elliptic curves; for supersingular curves, they reached a very high level of computational efficiency, and for ordinary curves, they found some counterparts of the nice features (e.g. distortion map) of the supersingular curves.

We now distinguish between the cases of supersingular and ordinary elliptic curves as follows:

- The main advantage of the supersingular curves is the existence of a distortion map defined as $\phi : E(\mathbb{F}_q) \to E(\mathbb{F}_{q^k})$. We now let $\mathbb{G}_1 = \mathbb{G}_2 = E(\mathbb{F}_q)$, the modified Tate pairing over supersingular elliptic curves will be then defined as follows:

  $$\text{For } P, Q \in E(\mathbb{F}_q), \hat{t}(P, Q) = t(P, \phi(Q))^{(q^k-1)/r}$$

  This is what we have previously categorized as symmetric pairings or Type–1 pairings. It has the advantage that $\hat{t}(P, Q) = \hat{t}(Q, P)$ which solves the hashing problem that will occur in the ordinary curve case. An additional advantage appears when dealing with fields of characteristic three, pairings will be much faster due to efficient tripling formulae.

[1] A divisor is just a notation for a finite multi-set of points [Gal12]. For a function f, a divisor (f) is a way to write down the intersection points (and their multiplicities) of f and a curve E.

[2] In 1986, Victor Miller described an algorithm for evaluating the Weil pairing on an algebraic curve. Although the paper has never been published, it is the basis of pairing based cryptography, and it had lots of follow on work in cryptography.

- In the ordinary elliptic curve case, there is no distortion map, and the pairings will be asymmetric since Q will be in $E(\mathbb{F}_{q^k})$. The modified Tate pairings over ordinary elliptic curves will be then defined as follows:

$$\text{For } P \in E(\mathbb{F}_q), Q \in E(\mathbb{F}_{q^k}), \hat{t}(P,Q) = t(P,Q)^{(q^k-1)/r}$$

  Barreto *et al.* [BKLS02, BLS04a] tried to find some counterparts to the nice features of the supersingular curves and came up with some ideas which improve the performance of the pairings. If r doesn't divide $q-1$, k is even and $x(Q) \in \mathbb{F}_{q^{k/2}}$ then we can use both the denominator elimination technique in Miller's algorithm and the twist technique.

Twist curve definition [Gal12] Let E be an elliptic curve over $\mathbb{K}$. A *twist* of E is an elliptic curve $\tilde{E}$ over $\mathbb{K}$ such that there is an isomorphism $\phi : E \to \tilde{E}$ over $\bar{\mathbb{K}}$ of pointed curves (such that $\phi(\mathcal{O}_E) = \mathcal{O}_{\tilde{E}}$).

Example of twist curves [BLS04a] Let E be given by $y^2 = x^3 + ax + b$ defined over $\mathbb{F}_{q^k}$, and consider its twist defined over $\mathbb{F}_{q^{k/2}}$ as $\tilde{E}(\mathbb{F}_{q^{k/2}}) : y^2 = x^3 + v^2ax + v^3b$ for some quadratic non-residue $v \in \mathbb{F}_{q^{k/2}}$. In F_{q^k}, v is a quadratic residue, which means the existence of an isomorphism $\Psi : \tilde{E}(\mathbb{F}_{q^{k/2}}) \to E(\mathbb{F}_{q^k})$ such that $\Psi(X,Y) = (v^{-1}X, (v\sqrt{v})^{-1}Y)$.

The advantage of using the twist curve is by performing the operations that do not use pairing (such as key generation and point transmission) use only arithmetic on $\mathbb{F}_{q^{k/2}}$. That was the case of a quadratic twist. Note that all elliptic curves have quadratic twists (i.e. degree 2). So far, for the higher-order twists we have two cases [BN06]:

- Curves that have $quartic$ twists: those with CM discriminant $= 1$, defined by $y^2 = x^3 + ax$.
- Curves that have $cubic$ and $sextic$ twists: those with CM discriminant $= 3$, defined by $y^2 = x^3 + b$.

3.6 Bilinear Diffie–Hellman Problem BDH and some of its variants

When cryptographers found out about the *attractive* relationship between bilinear mappings and elliptic curves, which are considered to be more efficient than finite fields, it happened that most of the research have relied on the symmetric pairing, namely Type-1. That's why the literature has lots of different notations and symbols, and sometimes they can easily get the reader confused of which is which. Below we give the definitions of computational and decisional bilinear Diffie-Hellman:

CBDH The Computational Bilinear Diffie–Hellman problem CBDH is as follows:
Given $g, g^a, g^b, g^c \in \mathbb{G}$.
Compute $e(g,g)^{abc}$.

DBDH The Decisional Bilinear Diffie–Hellman problem DBDH is as follows:
Given $g, g^a, g^b, g^c, g^z \in \mathbb{G}$.
Decide if $e(g,g)^{abc} = e(g,g)^z$.

In [SV07], they abstractly define the pairing problem that deals with groups $\mathbb{G}_1$ and $\mathbb{G}_2$ as $\mathbb{G}_i$ where $i, j, k \in \{1, 2\}$, and they define a pairing problem instance which is $\Gamma = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ where $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ denote groups of prime order q. g_1 and g_2 are two fixed generators of $\mathbb{G}_1$ and $\mathbb{G}_2$. Below, we present the three different types of BDH.

BDH$_{i,j,k}$: We define the BDH$_{i,j,k}$ problem to be the following:
Given g_i^a, g_j^b, and g_k^c, with $a, b, c \in \mathbb{F}_q$, compute $e(g_1, g_2)^{abc}$

One can easily notice that the original definition of BDH is the same as BDH$_{1,1,1}$.

coBDH$_{j,k}$: We define the coBDH$_{j,k}$ problem to be the following:
Given g_1^a, g_2^a, g_j^b, and g_k^c, with $a, b, c \in \mathbb{F}_q$, compute $e(g_1, g_2)^{abc}$.

BDH$^{\phi}_{i,j,k}$: We define the BDH$^{\phi}_{i,j,k}$ problem to be the following:

Given g_i^a, g_j^b, and g_k^c, with $a, b, c \in \mathbb{F}_q$, compute $e(g_1, g_2)^{abc}$. In this case, the adversary has access to an oracle which computes values under the isomorphism ϕ.

Notation:

$A \geq B$ means that problem B is no harder than problem A, which means that assumption A is weaker than assumption B (or B is a stronger assumption comparing to A). Moreover, if an adversary $\mathcal{F}$ can solve A, then $\mathcal{F}$ can be used as an Oracle in order to solve B. $A \equiv B$ means that the assumptions A and B have the same hardness level, i.e. $A \geq B$ and $B \geq A$.

Relationships between different BDH types

In [SV07], they give the following relationships between the different types of BDH:

1. if $i + j + k = i' + j' + k'$, then $\text{BDH}_{i,j,k} \equiv \text{BDH}_{j',j',k'}$. This means that any two problems that have two inputs from $\mathbb{G}_1$ and one from $\mathbb{G}_2$ and vice versa are equivalent. The same holds for $\text{BDH}^{\phi}_{i,j,k}$ and $\text{BDH}^{\phi}_{i',j',k'}$.

2. $i + j + k \leq i' + j' + k' \implies \text{BDH}^{\phi}_{i,j,k} \geq \text{BDH}^{\phi}_{j',j',k'}$. This can be easily checked. let's take the example where $\text{BDH}^{\phi}_{1,1,1} \geq \text{BDH}^{\phi}_{2,2,2}$. In fact the isomorphism ϕ will map all elements in $\mathbb{G}_2$ into elements in $\mathbb{G}_1$, and so the BDH problem will have all its inputs from $\mathbb{G}_1$, thus it can be solved using the $\text{BDH}^{\phi}_{1,1,1}$.

3. The trivial fact that says, $\text{BDH}_{i,j,k} \geq \text{BDH}^{\phi}_{i,j,k}$. If one can solve the BDH problem without the existence of ϕ, then he can solve it with its existence.

4. $\text{BDH}_{i,j,k} \geq \text{coBDH}_{j,k} \geq \text{BDH}^{\phi}_{2,j,k}$.

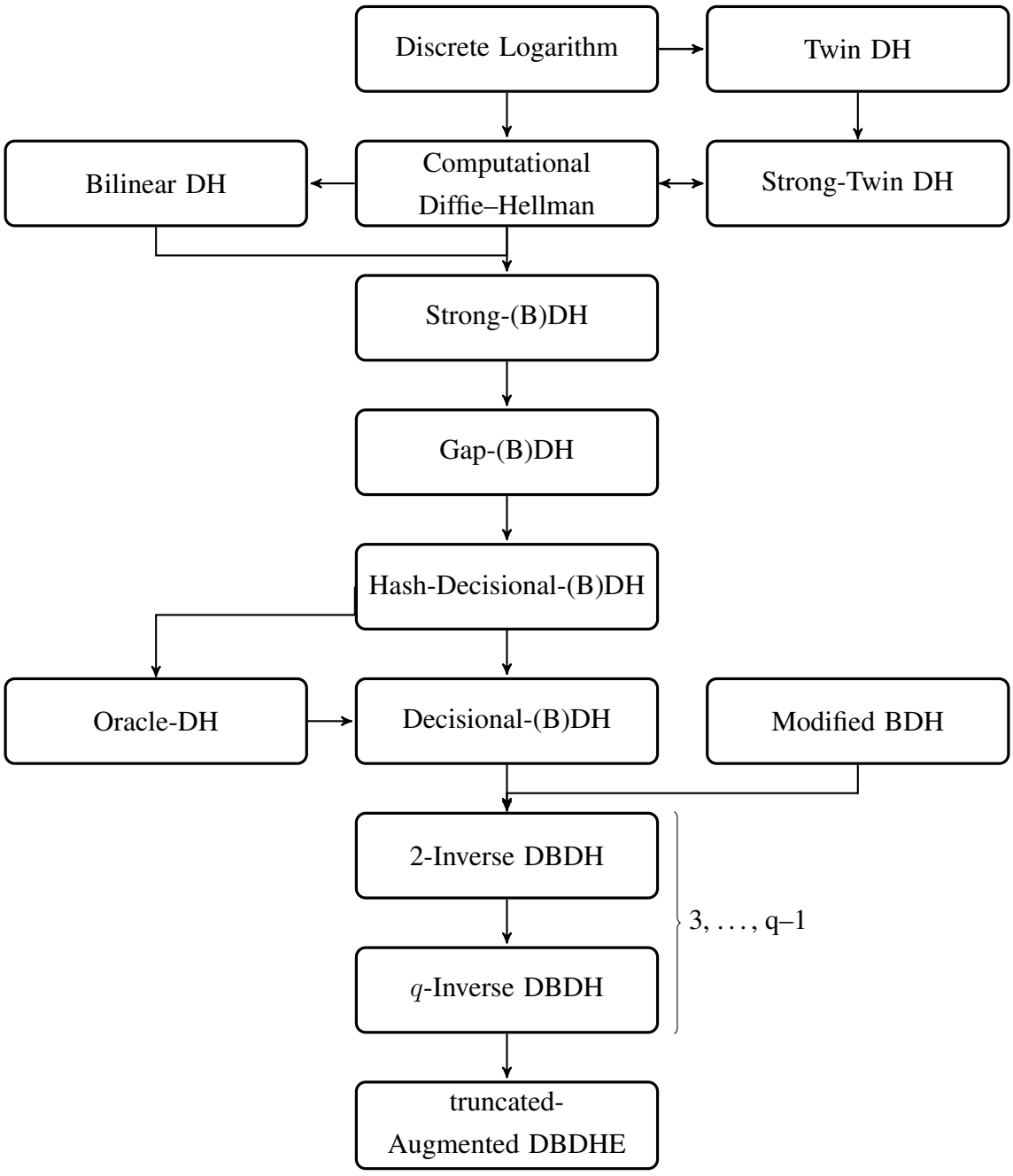
   For $\text{BDH}_{i,j,k} \geq \text{coBDH}_{j,k}$, it's easy to notice that $\text{coBDH}_{j,k}$ has all the input elements of $\text{BDH}_{i,j,k}$ plus one extra element so it can be easily solved using $\text{BDH}_{i,j,k}$ as an oracle.

   For $\text{coBDH}_{j,k} \geq \text{BDH}^{\phi}_{2,j,k}$, using the function ϕ one can get g_1^a from the given g_2^a, and then feed them to the $\text{coBDH}_{j,k}$ to compute $e(g_1, g_2)^{abc}$.

Next, we give a table of some of the existing hardness assumptions with their abbreviations, and a diagram of reductions between some of them. The details will be given in appendix A.

Diffie–Hellman and Bilinear Diffie–Hellman Variants		Page
CDH	Computational Diffie–Hellman	165
DDH	Decisional Diffie–Hellman	165
GDH	Generalized Diffie–Hellman	165
SCDH	Square Computational Diffie–Hellman	165
Inv–CDH	Inverse Computational Diffie–Hellman	165
DCDH	Divisible Computational Diffie–Hellman	165
ł-wDH or ł-DHI	l-weak Diffie–Hellman or Diffie–Hellman Inverse	166
q–SDH	q-Strong Diffie–Hellman	166
HDDH	Hash Diffie–Hellman	166
ODDH	Oracle Diffie–Hellman	166
SDH	Strong Diffie–Hellman	166
GapDH	Gap Diffie–Hellman	167
GapHDDH	Gap Hash Diffie–Hellman	167
DLIN	Decision Linear Diffie–Hellman	167
BDH	Computational Bilinear Diffie–Hellman	49
DBDH	decisional Bilinear Diffie–Hellman	49
l–BDHI	l–Bilinear Diffie–Hellman Inversion	169
l–wBDH	l–weak Bilinear Diffie–Hellman Inversion	169
BPI	Bilinear Pairing Inversion	169
l–BDHE	l–Bilinear Diffie–Hellman Exponent	169
TwinDH	Twin Diffie–Hellman	167
STwinDH	Strong Twin Diffie–Hellman	167
TwinDDH	Twin Decisional Diffie–Hellman	168
STwinDDH	Strong Twin Decisional Diffie–Hellman	168
STwinHDDH	Strong Twin Hashed Diffie–Hellman	168
TwinBDH	Twin Bilinear Diffie–Hellman	169
STwinBDH	Strong Twin Bilinear Diffie–Hellman	169
GapBDH	Gap Bilinear Diffie–Hellman	169
BDDH	modified Bilinear decisional Diffie–Hellman	170
q–BDDHI	q– Bilinear decisional Diffie–Hellman Inversion	170
truncated q–ABDHE	truncated q– Augmented Bilinear DH Exponent	170

3.7 Diagram of Reductions



3.8 Zero Knowledge Proofs (ZKP)

A non-mathematical introduction to zero-knowledge proofs is provided by [QQQ+90] in which they entertainingly narrate the secret of Ali Baba's legendary cave, and how it got rediscovered by Mick Ali later on. They explain the idea of the zero-knowledge property of such proofs in a very simple way. Originally, the idea of zero-knowledge proofs started with the fascinating work presented in [GMR89]. Prior to their work, the main focus was on the *soundness* of the proofs, i.e. to deal with a malicious Prover who attempts to fool the verifier into believing a false proof. In [GMR89], they tried to deal with malicious verifiers. What happens if we don't trust the verifier? The intention was to come up with some *convincing* proofs of some statements that reveal *nothing* beyond the validity of the assertion being proved. It's this contradictory nature that makes it appealing to cryptographers and therein lies their importance and vast applicability in cryptography; one always wanted to make sure that malicious parties are *surely* following certain predetermined protocols. Now, those malicious parties are forced to provide zero-knowledge proofs of the correctness of any internal secret action that they want to perform during the execution of the protocol, but without revealing their *actual* secrets.

Zero-knowledge proofs can be considered as a general class of protocols between two parties, called the prover P and the verifer V. Trivially, zero-knowledge proofs should allow a prover to prove to a verifier the validity of a certain assertion (Completeness). Moreover, prover may not fool a verifier to accept false assertion (Soundness). Both completeness and soundness must hold with high probability. The zero-knowledgeness property which is the most important property of ZKP assures that the proof doesn't reveal anything beyond the validity of such assertion/statement. The exciting result appears in [GMW91]; they prove that all languages in NP have zero-knowledge proof systems. Well, it does exist for the three-coloring problem[1], which is in the class *NP-complete*, and since any *NP*-problem can be translated into an instance of that problem, hence the existence of ZKPs for any NP problem.

Loosely speaking, to prove that a proof is zero knowledge we proceed as follows; we fix a prover P, and we show that whatever an *arbitrary* feasible adversary, i.e. a verifier V can compute after his interaction with P on an input (assertion) x, is *computationally indistinguishable* from what an arbitrary non-interactive feasible algorithm

[1] See [ZKP14] for an interactive zero knowledge 3-colorability demonstration.

can compute on his own given the same input x. The latter is what we call simulation. If one can simulate the interaction with a certain prover, i.e. without having access to the secret, and yet the result is computationally indistinguishable from the result of interacting with the prover, we conclude that the interaction doesn't leak any extra information beyond the validity of the assertion. In other words, if the verifier can't distinguish between the real case (where there is a secret) and simulated case (there is no secret), then the amount of information that he can extract is the same, and obviously doesn't depend on the "secret", hence the *zero-knowledgeness* of a proof system.

Let $R = \{(x, y) \subset X \times Y\}$ be an efficiently computable binary relation. For any pair $(x, y) \in R$, y is called the *statement* and x is the *witness* for y. Let L_R be the language consisting of statements in R i.e., $L_R = \{y \in Y : \exists x \in X, (x, y) \in R\}$

Completeness: A zero-knowledge proof is perfectly complete if all runs between honest prover and honest verifier are accepting. A zero knowledge protocol is ϵ_1-*incomplete* if for all $(x, y) \in R$, the interaction between an honest prover and an honest verifier fails with probability at most ϵ_1.

Soundness: A zero-knowledge proof is ϵ_2-*unsound* if an honest verifier accepts an incorrect input y with probability at most ϵ_2. An input y is *incorrect* if $(x, y) \notin R$ for all possible witnesses x.

Zero Knowledge: As defined in [GO94] and [GMR89], an interactive proof system for a language L_R is *zero-knowledge* if, for all probabilistic polynomial-time verifiers[1] V^*, there exists a probabilistic polynomial-time algorithm (simulator) S_{V^*} that on input y produces a probability distribution $S_{V^*}(y)$ such that $\{S_{V^*}(y)\}_{y \in L_R}$ and $\{< P(y), V^*(y) >\}_{y \in L_R}$ are polynomially indistinguishable. According to [GMR89], this version of zero-knowledge is often called *computational* zero-knowledge. The proof system is called *perfect* zero-knowledge if the aforementioned distributions are equal, whereas it's a *statistical* zero-knowledge if those distributions are statistically close.

Simulation Sound Zero Knowledge: Similar to the soundness property of proof systems, this states that with overwhelming probability, the prover should be incapable of convincing a polynomially bounded verifier of a false statement even after letting him see any number of simulated proofs of his choosing.

[1]Which can be thought of as a nondeterministic Turing Machine (TM) which randomly chooses between available transitions at each point according to some probability distribution.

Proofs vs. Arguments In zero-knowledge proof system, the proof remains valid even if an infinitely-powerful prover is involved [GMR89] whereas in a zero-knowledge arguments, it is required that only polynomially-bounded provers cannot cheat (except with negligible probability), given some complexity assumption [BCC88].

3.8.1 Σ Protocols

A protocol P is said to be a Σ-protocol [Sma13] for a relation R if:

- P is of a 3-move form, which means that the *statement* should be known to both *prover* and *verifier* ahead, the prover sends the verifier a *commitment a*, the verifier responds with a *challenge c* and then the prover sends his *response z*. To better understand the idea, we will give an example on how to prove the possession of a *secret* or *witness x*, for which $y = g^x$ is public. The protocol for a proof of knowledge of x goes as follows:
  - $P \to V : a = g^k$ for a random k,
  - $V \to P : c$
  - $P \to V : z = k + c \cdot x \pmod{q}$.

    The verifier now verifies that the prover knows the secret discrete logarithm x by verifying that: $a = g^z y^{-c}$.
- Completeness: if P, V follow the protocol on input y and private input x to P where $(x, y) \in R$, the verifier always accepts.
- Special Soundness: From any y and any pair of accepting conversations on input y, $(a, c_1, z_1), (a, c_2, z_2)$ where $c_1 \neq c_2$, one can efficiently compute x such that $(x, y) \in R$. This is sometimes called the *special soundness property*, which means it's indeed a proof of knowledge where the witness can be extracted. In the previous example, it's easy to see that the witness would be $x = \frac{z_2 - z_1}{c_2 - c_1} \pmod{q}$.
- Special Honest-Verifier Zero Knowledge: There exists a polynomial-time simulator S, which on input y and a random c outputs an accepting conversation of the form (a, c, z), with the same probability distribution as conversations between the honest P, V on input y. This property is called *special honest-verifier*

zero knowledge, HVZK. One can easily see that the verifier doesn't learn anything as he can simulate the whole transcript as follows:

- Generate a random value c (mod q).
- Compute $a = g^z y^{-c}$

Eventually, he outputs the transcript (a, c, z). Someone cannot tell the simulation of a transcript from a real transcript, and therefore the order in the interaction is what matters.[1]

3.8.2 Non-Interactive Zero Knowledge (NIZK) Proofs

Non-Interactive Zero Knowledge (NIZK), introduced by Blum, Feldman, and Micali in 1988 [BFM88], is a fundamental cryptographic primitive that has been used implicitly or explicitly in a majority of current cryptosystems. It mainly tries to remove the interaction between the prover and the verifier in zero-knowledge proof systems. Let R be an NP relation on pairs (x, y) with a corresponding language $\mathcal{L}_{\mathsf{R}} = \{y \mid \exists\ x\ s.t.(x, y) \in \mathsf{R}\}$. A NIZK proof system Π for a relation R is a tuple of algorithms ($\mathsf{NIZK.Setup}$, $\mathsf{NIZK.Prove}$, $\mathsf{NIZK.Verify}$, $\mathsf{NIZK.Extract}$, $\mathsf{NIZK.SimSetup}$, $\mathsf{NIZK.SimProve}$) defined as follows: $\mathsf{NIZK.Setup}$ takes as input a security parameter λ and outputs a reference string crs and an extraction key xk which allows for witness extraction. On input (crs, x, y), $\mathsf{NIZK.Prove}$ outputs a proof π if $\mathsf{R}(x, y) = 1$. On input (crs, y, π), $\mathsf{NIZK.Verify}$ outputs 1 if π is a valid proof that $y \in \mathcal{L}_{\mathsf{R}}$, and 0 otherwise. $\mathsf{NIZK.Extract}$ outputs the witness x from a valid proof π. Finally, $\mathsf{NIZK.SimSetup}$ outputs a simulated reference string $\mathsf{crs}_{\mathsf{sim}}$ and a trapdoor tr, which is used by $\mathsf{NIZK.SimProve}$ to simulate proofs without a witness. Below we define the properties that a non-interactive zero-knowledge proof (NIZK) system can enjoy. Note that a NIZK system has to be complete, sound and zero knowledge whereas the need for the other properties depends on the case for which the NIZK is being used:

- **Perfect Completeness:** $\forall\lambda \in \mathbb{N}$, $\forall(x, y) \in \mathsf{R}$, we have

$$\Pr\left[(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^\lambda); \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, y) : \mathsf{Verify}(\mathsf{crs}, y, \pi) = 1\right] = 1 \ .$$

[1] In other words, a verifier who succeeds in extracting information from a real proof must also be able to extract information from a simulated one where no information is available in the first place! This actually leads to a contradiction, and proves that the protocol can't actually leak sensitive information in either situation.

- **Soundness:** $\forall \lambda \in \mathbb{N}$, $\forall y \notin \mathcal{L}_{\mathsf{R}}$, we have for all adversaries $\mathcal{F}$

$$\Pr\left[(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^\lambda); \pi \leftarrow \mathcal{F}(\mathsf{crs}, y) : \mathsf{Verify}(\mathsf{crs}, y, \pi) = 1\right] \leq 2^{-\lambda} \ .$$

  If the above probability is 0, we say the system has *perfect soundness*.

- **Knowledge Extraction:** A proof system is a *Proof of Knowledge* if there exists an efficient extractor algorithm Extract which can extract the witness from any proof the adversary outputs. Note that if a proof system is a proof of knowledge then it is sound. More formally, for all adversaries $\mathcal{F}$, we have

$$\begin{aligned}\Pr\big[(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^\lambda); (y, \pi) \leftarrow \mathcal{F}(\mathsf{crs}); x \leftarrow \mathsf{Extract}(\mathsf{crs}, \mathsf{xk}, y, \pi) \\ : \mathsf{Verify}(\mathsf{crs}, y, \pi) = 0 \text{ OR } (x, y) \in \mathsf{R}\big] \leq 1 - \nu(\lambda) \ .\end{aligned}$$

  Where $\nu(\lambda)$ is a negligible function. If the above probability is 1, we say the system has *perfect knowledge extraction*.

- **Witness Indistinguishability:** The system is *witness indistinguishable* if for all PPT adversaries $\mathcal{F}$, we have

$$\Pr\left[\begin{array}{l}(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^\lambda); (\mathsf{st}_{\mathsf{find}}, y, x_0, x_1) \leftarrow \mathcal{F}_{\mathsf{find}}(\mathsf{crs}); \\ b \leftarrow \{0,1\}; \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x_b, y); b^* \leftarrow \mathcal{F}_{\mathsf{guess}}(\mathsf{st}_{\mathsf{find}}, \pi) \\ : (x_0, y) \in \mathsf{R} \ \wedge \ (x_1, y) \in \mathsf{R} \ \wedge \ b = b^*\end{array}\right] = \frac{1}{2} + \nu(\lambda)$$

  If $\nu(\lambda) = 0$, we say the system has *perfect witness indistinguishability*.

- **Zero-Knowledge:** The system is *zero-knowledge* if $\forall (x, y) \in \mathsf{R}$, we have for all PPT adversaries $\mathcal{F}$

$$\begin{aligned}&\Pr\left[(\mathsf{crs}_{\mathrm{sim}}, \mathsf{tr}) \leftarrow \mathsf{SimSetup}(1^\lambda) : \mathcal{F}^{\mathsf{Sim}(\mathsf{crs}_{\mathrm{sim}}, \mathsf{tr}, \cdot, \cdot)}(\mathsf{crs}_{\mathrm{sim}}) = 1\right] \\ &\quad \approx \Pr\left[(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^\lambda) : \mathcal{F}^{\mathsf{Prove}(\mathsf{crs}, \cdot, \cdot)}(\mathsf{crs}) = 1\right],\end{aligned}$$

  where $\mathsf{Sim}(\mathsf{crs}_{\mathrm{sim}}, \mathsf{tr}, x, y)$ outputs $\mathsf{SimProve}(\mathsf{crs}_{\mathrm{sim}}, \mathsf{tr}, y)$ if $(x, y) \in \mathsf{R}$ or $\perp$ otherwise.

- **Simulation-Soundness:**[1] The system is *simulation-sound* [Sah99] if the adver-

[1] Note that in [Sah99, GMY03], they show how to add a *strong one time signature* scheme to a

sary cannot produce a proof for a false statement even after seeing simulated proofs for possibly false statements. Formally, for all PPT adversaries $\mathcal{F}$ we have

$$Pr\left[\begin{array}{l}(\mathsf{crs}_{\mathsf{Sim}},\mathsf{tr})\leftarrow\mathsf{SimSetup}(1^\lambda);(\pi^*,y^*)\leftarrow\mathcal{F}^{\mathsf{SimProve}(\mathsf{crs}_{\mathsf{Sim}},\mathsf{tr},\cdot)}(\mathsf{crs}_{\mathsf{Sim}})\\ :(\pi^*,y^*)\notin\mathcal{Q}\ \wedge\ \mathsf{Verify}(\mathsf{crs},y,\pi)=1\ \wedge\ y\notin\mathcal{L}\end{array}\right]\approx 0$$

If we limit the number of queries to one, we call the system *one-time simulation-sound*.

- **Simulation Sound Extractability:**[1] By combining simulation-soundness and knowledge extraction, we get *Simulation-Sound Extractable Proofs* [Gro06]. This requires that we can extract a witness from any proof the adversary outputs even after seeing simulated proofs. More formally, (here we abuse the notation and assume that SimSetup now also outputs the extraction key xk), we have for all PPT adversaries $\mathcal{F}$ that

$$Pr\left[\begin{array}{l}(\mathsf{crs}_{\mathsf{Sim}},\mathsf{tr},\mathsf{xk})\leftarrow\mathsf{SimSetup}(1^\lambda);\\ (y,\pi)\leftarrow\mathcal{F}^{\mathsf{SimProve}(\mathsf{crs}_{\mathsf{Sim}},\mathsf{tr},\cdot)}(\mathsf{crs},\mathsf{xk});\\ x\leftarrow\mathsf{Extract}(\mathsf{crs},\mathsf{xk},y,\pi):(y,\pi)\notin\mathcal{Q}\ \wedge\ \mathsf{Verify}(\mathsf{crs},y,\pi)=1\\ \wedge\,(x,y)\notin\mathsf{R}\end{array}\right]\leq\nu(\lambda)$$

Now, we present two versions of NIZK systems; one in the random oracle model where the idea is to use a hash function H that works as a random oracle in order to avoid the interactivity in the proofs. This NIZK system [FS87] is simulation sound extractable [FKMV12]. The second one is in the standard model that needs a common reference string between parties but doesn't use ideal hash functions (random oracle). This NIZK system [GS08] is not simulation sound extractable [GS08].

3.8.2.1 Fiat-Shamir Heuristic

In [FS87], they give a method to avoid the interactivity in the ZK system, i.e. non-iteractive ZK arguments. To see how this works in a simple example, we can take the previous example of possession of a secret discrete logarithm; we can avoid the

regular NIZK system (i.e. it is not simulation sound) to eventually get a *simulation-sound* NIZK.

[1] Also called Non-malleability in the literature, note that a non-malleable NIZK is also simulation sound [GMY03].

second step, i.e. the verifier sends the challenge to the prover by hashing the value of the commitment to get c, $c \leftarrow H(a)$, so the verifier can apply the hash function on the commitment being sent by the prover in order to get the challenge, and use it later on in the verification algorithm.

3.8.2.2 GS-proofs

In 2007, Groth and Sahai [GS08] proposed new non-interactive witness indistinguishable (WI) and zero knowledge (ZK) proofs based on bilinear maps, for which they gave three instantiations according to the admitted assumption. While the zero knowledge property assures that a verifier can't learn anything about the witness of a statement, the witness indistinguishablity assures that, in the case that an NP assertion has more than one witness, if a prover provides a certain proof that uses one of these witnesses, the verifier can't tell which witness is being used. It's clear that zero knowledgeness implies witness indistinguishability. The first instantiation is based on the subgroup decision problem, which is; suppose we have a composite order bilinear group $(n, \mathbb{G}, G_T, e, P)$ where $n = pq$. Then $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q$, where $\mathbb{G}_p, \mathbb{G}_q$ are the subgroups of prime order p and q respectively. Boneh, Goh and Nissim introduced the *subgroup decision* assumption, which claims that it is hard to distinguish a random element from $\mathbb{G}$ from a random element from $\mathbb{G}_q$. It's the most efficient instantiation, but it works with Type-1 bilinear maps [1]. The second instantiation is based on the XDH assumption. The Symmetric XDH assumption says that the DDH problem is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$. The third instantiation is based on the decisional linear Diffie-Hellman problem DLIN, which assumes that, given $u, v, h \in \mathbb{G}$ and $u^a, v^b, h^c \in \mathbb{G}$, it's hard to tell if $c = a + b$ or it's a random in Z_p. The intuition of this NIZK is the following [GS08]; to prove that *z is the encryption of m under the product of the two keys x and y*, we encode it this way; *c is the encryption of the value committed to in d under the product of the two keys committed to in a and b.*

Assume that we have the following bilinear maps $f : A_1 \times A_2 \rightarrow A_3$ and $F : B_1 \times B_2 \rightarrow B_3$, and we have a set of simultaneous equations of the following types: Pairing Product Equation (PPE), Multi-scalar multiplication Equations (MME) in A_1

[1] As we mentioned earlier, Type-1 pairing is not recommended any more because of the fact that several recent attacks have been realised to solve the discrete logarithm problem DLog in the multiplicative finite fields of small characteristic (2 or 3), and the nature of the Elliptic curves used in Type-1 pairings (super singular curves) necessitate the use of such fields.

(respectively in A_2), and Quadratic Equations in Z_n. The idea is to construct a witness-indistinguishable/Zero knowledge proofs for the simultaneous satisfiability of a set of such equations. In the common reference string that will be provided to the verifier, a description of the following maps will be given: $\{\iota_i : B_i \to A_i\}_i$ which are the commitment schemes and $\{\rho_i : A_i \to B_i\}_i$. The fact that the commitment schemes are homomorphic will give the following results:

- $\forall x \in A_1, \forall y \in A_2 : F(\iota_1(x), \iota_2(y)) = \iota_3(f(x, y))$
- $\forall x \in B_1, \forall y \in B_2 : f(\rho_1(x), \rho_2(y)) = \rho_3(F(x, y))$

There are two settings when it comes to producing the common reference string (crs); in the first setting, the keys used in ι_i are *hiding keys*, which means that ι is perfectly hiding (which offers the WI/ZK), whereas in the second setting, the keys used in ι are *binding* (which offers witness extractibility). The main idea of GS proofs is the following: the two settings that offer the soundness and the WI/ZK alternatively, are computationally indistinguishable based on the assumption that the distributions of the hiding and the biding keys are computationally indistinguishable.

Syntax and Equations The language for these proofs is of the form

$$\mathcal{L} := \{\mathsf{statement\ st} \mid \exists\ \mathsf{witness}\ \underline{\mathsf{w}} : E(\mathsf{st}, \underline{\mathsf{w}})\ \mathsf{holds}\ \}$$

$E(\mathsf{statement}, \cdot)$ could be one of the three types that we will describe below, where these equations are defined by the public values $\{A_i, T \in \mathbb{G},\ a_i, b_i, \Gamma = (\gamma_{ij}) \in \mathrm{Mat}_{n\times m}, t \in \mathbb{Z}_p,\ t_T \in \mathbb{G}_T\}$, whereas the witness $\subset \{X_1, \ldots, X_m, Y_1, \ldots, Y_n \in \mathbb{G},\ x_1, \ldots, x_m, y_1, \ldots, y_n \in \mathbb{Z}_p\}$, which we underline for simplicity.

- **Pairing Product Equation (PPE):**

$$\prod_{i=1}^{n} e(A_i, \underline{Y_i}) \cdot \prod_{i=1}^{m} e(\underline{X_i}, B_i) \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{ij}} = t_T{}^{1}$$

- **Multi-Scalar Multiplication Equation (MSME) in $\mathbb{G}_1$:**[2]

[1] In the NIZK, t_T should be either 1 or $\prod_{i=1}^{n} e(P_i, Q_i)$, for some known P_i, Q_i.

[2] Similarly, we have a MSME in $\mathbb{G}_2$.

$$\prod_{i=1}^{n} A_i^{\underline{y_i}} \prod_{i=1}^{m} \underline{X_i}^{b_i} \prod_{i=1}^{m} \prod_{j=1}^{n} \underline{X_i}^{\gamma_{ij}\underline{y_j}} = T$$

- **Quadratic Equation (QE) in $\mathbb{Z}_p$:**

$$\sum_{i=1}^{n} a_i \underline{y_i} + \sum_{i=1}^{m} \underline{x_i} b_i + \sum_{i=1}^{m} \sum_{j=1}^{n} \underline{x_i} \underline{y_j} = t$$

The GS proof system is formally defined by the following tuple of algorithms

$$\mathsf{GS} = (\{\mathsf{GSSetup}, \mathsf{GSProve}, \mathsf{GSVerify}, \mathsf{GSExtract}\}, \{\mathsf{GSSimSetup}, \mathsf{GSSimProve}\})$$

Soundness Setting:

- GSSetup : Takes as input the description of a bilinear group $\mathcal{P}$ and outputs a *soundness* reference string crs and an extraction key xk.
- GSProve : Takes as input crs, a set of equations statement and a witness, and outputs a proof Ω for the satisfiability of the equations.
- GSVerify : Given crs, a set of equations and a proof Ω, it outputs 1 if the proof is valid, and else 0.
- GSExtract : Takes as input a soundness crs, the extraction key xk and a valid proof Ω, and outputs the witness used for the proof.

Witness indistinguishability /Zero knowledge (WI/ZK) Setting:

- GSSimSetup : On input a bilinear group description $\mathcal{P}$, outputs a *simulation* reference string $\mathsf{crs}_{\mathrm{Sim}}$ and a trapdoor key tr that allows us to simulate proofs.
- GSSimProve : Takes $\mathsf{crs}_{\mathrm{Sim}}$, a statement and the trapdoor tr and produces a simulated proof Ω_{Sim} without a witness.

The main assumption on which GS proofs rely is that the distributions of strings crs and $\mathsf{crs}_{\mathrm{Sim}}$ are computationally indistinguishable. In other words, the distributions of the hiding keys and the binding keys of the commitments schemes used respectively

in the *WI/ZK setting* and *soundness setting* are computationally indistinguishable, and therefore the simulated proofs are indistinguishable from proofs output by GSProve.

The proof system has prefect completeness, perfect soundness, composable witness-indistinguishability or composable zero-knowledge[GS08] [1].

We will give two examples to clarify the way GS proofs work. The first one is the following:

Example 3. *(As given in [CM14]) Let $A, B \in \mathbb{G}_1$, and $t \in \mathbb{G}_T$. We will give a NIWI proof of knowledge of $Y_1, Y_2 \in \mathbb{G}_2$, based on the DLIN assumption, such that:*

$$e(A, \underline{Y_1}) \cdot e(B, \underline{Y_2}) = t$$

- ***Setup:*** *Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$*
- ***Common Reference String*** crs*:*
  - *Let $H \in \mathbb{G}_2$, $a, b, i, j \in \mathbb{Z}_p$.*
  - *$U = aH, V = bH, I = iU, J = jV, K = (i + j)H$*
  - *$\mathsf{crs} = (U, V, I, J, K)$*
- ***Commitments:***
  - *Select $s_{11}, s_{12}, s_{13}, s_{21}, s_{22}, s_{23} \in \mathbb{Z}_p$*
  - *Compute:*

$$\begin{aligned}
d_{11} &= s_{11}U + s_{13}I, \\
d_{12} &= s_{12}V + s_{13}J, \\
d_{13} &= Y_1 + s_{11}H + s_{12}H + s_{13}K, \\
d_{21} &= s_{21}U + s_{23}I, \\
d_{22} &= s_{22}V + s_{23}J, \\
d_{23} &= Y_2 + s_{21}H + s_{22}H + s_{23}K
\end{aligned}$$

[1] As defined in [GS08], In composable zero-knowledge, the real crs and the simulated $\mathsf{crs}_{\mathsf{Sim}}$ are computationally indistinguishable. Moreover, the adversary, given access to the secret simulation key tr , still cannot distinguish real proofs from simulated proofs on a simulated $\mathsf{crs}_{\mathsf{Sim}}$, (See [GS08] for more details).

– *The commitment is:* $d = (d_{11}, d_{12}, d_{13}, d_{21}, d_{22}, d_{23})$.

- ***Proof:***

  – *Compute*

$$\theta_1 = s_{11}A + s_{21}B$$
$$\theta_2 = s_{12}A + s_{22}B$$
$$\theta_3 = s_{13}A + s_{23}B$$

  – *The proof is* $\theta = (\theta_1, \theta_2, \theta_3)$.

- ***Verification***

  Check that:

  – $\theta_1, \theta_2, \theta_3 \in \mathbb{G}_1, d_{11}, d_{12}, d_{13}, d_{21}, d_{22}, d_{23} \in \mathbb{G}_2$,

  – $e(A, d_{11}) \cdot e(B, d_{21}) = e(\theta_1, U) \cdot e(\theta_3, I)$

  – $e(A, d_{12}) \cdot e(B, d_{22}) = e(\theta_2, V) \cdot e(\theta_3, J)$

  – $e(A, d_{13}) \cdot e(B, d_{23}) = e(\theta_1, H) \cdot e(\theta_2, H) \cdot e(\theta_3, K) \cdot t$

Example 4. *The second example is the following. The equation is*

$$PPE(X, Y) : e(A, \underline{Y}) \cdot e(\underline{X}, B) \cdot e(\underline{X}, \underline{Y}) = 1 \tag{3.1}$$

determined by $A \in \mathbb{G}_1, B \in \mathbb{G}_2$. *The prover needs to show that he knows of witnesses* X *and* Y *that they satisfy the equation PPE. We chose the RHS of the equation to be one so that it's possible to give a NIZK according to the restrictions that Groth-Sahai have for the PPE equations. Roughly speaking, the prover will commit to the witnesses and give them away, along with some extra information (proofs) so that the verifier can check on the satisfiability of* larger *version of the equation PPE. Analogously to the pairing* e, *we define another bilinear map* E *that is suitable for the commitments: Let* $E : \mathbb{G}_1^2 \times \mathbb{G}_2^2 \to \mathbb{G}_T^4$, *with*

$$E^1\left((\lambda_1, \lambda_2), (\omega_1, \omega_2)\right) = \begin{pmatrix} e(\lambda_1, \omega_1) & e(\lambda_1, \omega_2) \\ e(\lambda_2, \omega_1) & e(\lambda_2, \omega_2) \end{pmatrix}$$

We can extend E so it can deal with vectors of pairs of group elements. The generalized form is $E^m : \mathbb{G}_1^{m\times 2} \times \mathbb{G}_2^{m\times 2} \to \mathbb{G}_T^4$. In this sense, E can be viewed as a special case of E^m, i.e, E^1. In our example we are going to need E^2, so one can think of it as

$$E^2\left(\begin{pmatrix}\lambda_{11} & \lambda_{12}\\ \lambda_{21} & \lambda_{22}\end{pmatrix}, \begin{pmatrix}\omega_{11} & \omega_{12}\\ \omega_{21} & \omega_{22}\end{pmatrix}\right)$$

$$= \begin{pmatrix} e(\lambda_{11},\omega_{11})\cdot e(\lambda_{21},\omega_{21}) & e(\lambda_{11},\omega_{12})\cdot e(\lambda_{21},\omega_{22}) \\ e(\lambda_{12},\omega_{11})\cdot e(\lambda_{22},\omega_{21}) & e(\lambda_{12},\omega_{12})\cdot e(\lambda_{22},\omega_{22}) \end{pmatrix}$$

Since we are going to use the second instantiation of the GS proofs, which is based on the SXDH assumption (it states that DDH is hard in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$), we will define the commitment scheme that they use in this instantiation along with the maps ι_1, ι_2 and ι_T where $\iota_i : \mathbb{G}_i \to \mathbb{G}_i^2$ such that $\iota_i(g) = (0, g)$ and $\iota_T : \mathbb{G}_T \to \mathbb{G}_T^4$ where $\iota(t_T) = \begin{pmatrix}1 & 1\\ 1 & t_T\end{pmatrix}$. For the commitment scheme, we choose $\alpha_1, \alpha_2, t_1, t_2 \leftarrow \mathbb{Z}_p$ so that the commitment keys (ck) are $\{\vec{u_1}, \vec{u_2}\}$ for the group elements in $\mathbb{G}_1$ and $\{\vec{v_1}, \vec{v_2}\}$ for $\mathbb{G}_2$ where $\vec{u_1} = (g_1, g_1^{\alpha_1}), \vec{u_2} = (g_1^{t_1}, g_1^{t_1\alpha_1}), \vec{v_1} = (g_2, g_2^{\alpha_2}), \vec{v_2} = (g_2^{t_2}, g_2^{t_2\alpha_2})$. According to the GS paper, let $\rho_1 = r_1 + t_1 r_2$, $\rho_2 = s_1 + t_2 s_2$, $R = (r_1, r_2) \leftarrow \mathbb{Z}_p$ and $S = (s_1, s_2) \leftarrow \mathbb{Z}_p$, then we commit to X and Y as follows:

- $c = \mathsf{Commit}(ck, X, R) = (g_1^{r_1} g_1^{t_1 r_2}, X g_1^{\alpha_1 r_1} g_1^{\alpha_1 t_1 r_2}) = (g_1^{\rho_1}, X g_1^{\alpha_1 \rho_1})$.
- $d = \mathsf{Commit}(ck, Y, S) = (g_2^{s_1} g_2^{t_2 s_2}, Y g_2^{\alpha_2 s_1} g_2^{\alpha_2 t_2 s_2}) = (g_2^{\rho_2}, Y g_2^{\alpha_2 \rho_2})$.

Based on equation 3.1, we will write another equation where the verifier is able to verify that the commitments to the witnesses of equation 3.1 satisfy it. The new equation is[1]

$$E^1(\iota_1(A), d) \odot E^1(c, \iota_2(B)) \odot E^1(c, \Gamma d) = \iota_T(1) \odot E^2(\vec{U}, \phi) \odot E^2(\theta, \vec{V}) \quad (3.2)$$

Where $\vec{U} = \begin{pmatrix}\vec{u_1}\\ \vec{u_2}\end{pmatrix}$ and $\vec{V} = \begin{pmatrix}\vec{v_1}\\ \vec{v_2}\end{pmatrix}$. For the verifier to be able to check on the satisfaction of equation 3.2, the prover still needs to compute ϕ and θ to send them along with the commitments, knowing that the maps ι_1, ι_2 and ι_T are parts of the public parameters.

[1] $\odot$ denotes component-wise multiplication

ϕ and θ. The two parts of the proof, ϕ and θ can be written as follows:[1]

- $\phi = R^\top \iota_2(B) + R^\top \Gamma \iota_2(Y) + (R^\top \Gamma S - T^\top)\vec{V}$.
- $\theta = S^\top \iota_1(A) + S^\top \Gamma^\top \iota_1(X) + T\vec{U}$.

In our example, $\Gamma = 1$. Note that, originally in the GS paper they used addition as the group law since we usually work over groups of points on elliptic curves, but for simplicity, we will use the multiplication as the group law, therefore

$$\begin{aligned}\phi &= \begin{pmatrix} v_{11}^{r_1 s_1} v_{21}^{r_1 s_2} & (BY)^{r_1} v_{12}^{r_1 s_1} v_{22}^{r_1 s_2} \\ v_{11}^{r_2 s_1} v_{21}^{r_2 s_2} & (BY)^{r_2} v_{12}^{r_2 s_1} v_{22}^{r_2 s_2} \end{pmatrix} \\ &= \begin{pmatrix} g_2^{r_1(s_1+s_2 t_2)} & (BY)^{r_1} g_2^{r_1 \alpha_2 (s_1+s_2 t_2)} \\ g_2^{r_2(s_1+s_2 t_2)} & (BY)^{r_2} g_2^{r_2 \alpha_2 (s_1+s_2 t_2)} \end{pmatrix} \\ &= \begin{pmatrix} g_2^{r_1 \rho_2} & (BY)^{r_1} g_2^{r_1 \alpha_2 \rho_2} \\ g_2^{r_2 \rho_2} & (BY)^{r_2} g_2^{r_2 \alpha_2 \rho_2} \end{pmatrix}\end{aligned}$$

and

$$\theta = \begin{pmatrix} 1 & (AX)^{s_1} \\ 1 & (AX)^{s_2} \end{pmatrix}$$

Verification *On input $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \iota_1, \iota_2, \iota_T, \vec{U}, \vec{V}, \phi, \theta, c, d)$ the verifier should be able to check on the satisfaction of equation 3.2 by simply evaluating E^1 and E^2 on both sides where their inputs are all given.*

[1] Usually, two extra terms are added to ϕ and θ that are $-T^\top \vec{V}$ and $T\vec{U}$ where $T \leftarrow \text{Mat}_{2\times 2}(\mathbb{Z}_p)$ but they always cancel each other, so we can safely omit them in the computation.

For the LHS of equation (3.2):

$$
\begin{aligned}
&E^1(\iota_1(A), d) \odot E^1(c, \iota_2(B) \odot E^1(c, d) \\
&= E^1((0, A), (g_2^{\rho_2}, Y g_2^{\alpha_2\rho_2})) \odot E^1((g_1^{\rho_1}, X g_1^{\alpha_1\rho_1}), (0, B)) \odot \\
&E^1((g_1^{\rho_1}, X g_1^{\alpha_1\rho_1}), (g_2^{\rho_2}, Y g_2^{\alpha_2\rho_2})) \\
&= \begin{pmatrix} 1 & 1 \\ e(A, g_2^{\rho_2}) & e(A, Y g_2^{\alpha_2\rho_2}) \end{pmatrix} \odot \begin{pmatrix} 1 & e(g_1^{\rho_1}, B) \\ 1 & e(X g_1^{\alpha_1\rho_1}, B) \end{pmatrix} \\
&\odot \begin{pmatrix} e(g_1^{\rho_1}, g_2^{\rho_2}) & e(g_1^{\rho_1}, Y g_2^{\alpha_2\rho_2}) \\ e(X g_1^{\alpha_1\rho_1}, g_2^{\rho_2}) & e(X g_1^{\alpha_1\rho_1}, Y g_2^{\alpha_2\rho_2}) \end{pmatrix} \\
&= \begin{pmatrix} 1 & e(g_1^{\rho_1}, B) \\ e(A, g_2^{\rho_2}) & e(A, Y g_2^{\alpha_2\rho_2}) \cdot e(X g_1^{\alpha_1\rho_1}, B) \end{pmatrix} \\
&\odot \begin{pmatrix} e(g_1^{\rho_1}, g_2^{\rho_2}) & e(g_1^{\rho_1}, Y g_2^{\alpha_2\rho_2}) \\ e(X g_1^{\alpha_1\rho_1}, g_2^{\rho_2}) & e(X g_1^{\alpha_1\rho_1}, Y g_2^{\alpha_2\rho_2}) \end{pmatrix} \\
&= \begin{pmatrix} e(g_1^{\rho_1}, g_2^{\rho_2}) & e(g_1^{\rho_1}, BY g_2^{\alpha_2\rho_2}) \\ e(AX g_1^{\alpha_1\rho_1}, g_2^{\rho_2}) & e(X g_1^{\alpha_1\rho_1}, B) e(AX g_1^{\alpha_1\rho_1}, Y g_2^{\alpha_2\rho_2}) \end{pmatrix}
\end{aligned}
$$

For the RHS of the equation (3.2):

$$
\begin{aligned}
&\iota_T(1) \odot E^2(\vec{U}, \phi) \odot E^2(\theta, \vec{V}) \\
&= \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \odot E^2\left(\begin{pmatrix} g_1 & g_1^{\alpha_1} \\ g_1^{t_1} & g_1^{t_1\alpha_1} \end{pmatrix}, \begin{pmatrix} g_2^{r_1\rho_2} & (BY)^{r_1} g_2^{r_1\alpha_2\rho_2} \\ g_2^{r_2\rho_2} & (BY)^{r_2} g_2^{r_2\alpha_2\rho_2} \end{pmatrix} \right) \\
&\odot E^2\left(\begin{pmatrix} 1 & (AX)^{s_1} \\ 1 & (AX)^{s_2} \end{pmatrix}, \begin{pmatrix} g_2 & g_2^{\alpha_2} \\ g_2^{t_2} & g_2^{t_2\alpha_2} \end{pmatrix} \right)
\end{aligned}
$$

$$
\begin{aligned}
&= \begin{pmatrix} e(g_1, g_2)^{\rho_1\rho_2} & e(g_1, BY)^{\rho_1} e(g_1, g_2)^{\alpha_2\rho_2\rho_1} \\ e(g_1, g_2)^{\alpha_1\rho_1\rho_2} & e(g_1, g_2)^{\alpha_1\alpha_2\rho_1\rho_2} e(g_1, BY)^{\alpha_1\rho_1} \end{pmatrix} \\
&\odot \begin{pmatrix} 1 & 1 \\ e((AX)^{s_1}, g_2) e((AX)^{s_2}, g_2^{t_2}) & e((AX)^{s_1}, g_2{}^{\alpha_2}) e((AX)^{s_2}, g_2^{t_2\alpha_2}) \end{pmatrix} \\
&= \begin{pmatrix} e(g_1, g_2)^{\rho_1\rho_2} & e(g_1, BY)^{\rho_1} e(g_1, g_2)^{\alpha_2\rho_2\rho_1} \\ e(g_1, g_2)^{\alpha_1\rho_1\rho_2} & e(g_1, g_2)^{\alpha_1\alpha_2\rho_1\rho_2} e(g_1, BY)^{\alpha_1\rho_1} \end{pmatrix} \\
&\odot \begin{pmatrix} 1 & 1 \\ e(AX, g_2)^{\rho_2} & e(AX, g_2)^{\rho_2\alpha_2} \end{pmatrix} \\
&= \begin{pmatrix} e(g_1, g_2)^{\rho_1\rho_2} & e(g_1^{\rho_1}, BY g_2^{\alpha_2\rho_2}) \\ e(AX g_1^{\alpha_1\rho_1}, g_2^{\rho_2}) & e(g_1, g_2)^{\alpha_1\alpha_2\rho_1\rho_2} e(g_1, BY)^{\alpha_1\rho_1} e(AX, g_2)^{\rho_2\alpha_2} \end{pmatrix}
\end{aligned}
$$

One can easily see that the first three elements of the matrices in the LHS and RHS match, and that LHS$_{2,2}$/RHS$_{2,2}$ = PPE(X, Y) which is equal to 1 as per our example. Therefore, equation (3.2) verifies correctly.

3.9 Multilinear Maps

A more general version of the bilinear maps, i.e multilinear Maps, has been considered one of the open problems for almost a decade. Finding useful multilinear maps was considered by Boneh and Silverberg (in 2002 [BS02]) as an almost impossible mission within the realm of algebraic geometry (Where they also showed how helpful multilinear maps could be to Broadcast Encryption). Surprisingly, Grag, Gentry, and Halvei, using ideal lattices, announced a promising achievement by finding something

that is *morally* equivalent to multilinear maps, i.e instead of having a function

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \times \cdots \times \mathbb{G}_n \to \mathbb{G}_T,$$

they define a set of bilinear maps as follows;

$$E = \{e_{i,j}; e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j} | i, j \geq 1, i + j \leq k\}$$

We let g_i be a canonical generator of $\mathbb{G}_i$, the map $e_{i,j}$ satisfies the following relation;

$$e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab} : \forall a, b \in \mathbb{Z}_p$$

Despite the intensive cryptanalysis that the authors have done in their paper, they themselves mentioned that the development and cryptanalysis of this tool are still at a nascent stage. Nevertheless, this result is an exciting opportunity to study new constructions using the Multilinear maps abstraction. Subsequent to their paper, another *practical* Multilinear scheme, this time over integers, has been built by Coron, Lepoint and Tibouchi [CLT13]. Note that this scheme showed serious vulnerabilities against *Zeroizing Attack* as presented in [CHL$^+$14]. The attack relies on an adaptation of the zeroizing attack against the Garg, Gentry and Halevi (GGH) candidate multilinear maps [GGH13a]. Some countermeasures have been introduced to mitigate against this attack [BWZ14], but as shown in [CLT14], the attack seems, so far, to stay valid albeit the adoption of those countermeasures.

In a nutshell, here are the two constructions:

- The first construction [GGH13a]:

  It works in the polynomial ring $R = \mathbb{Z}[X]/(X^n + 1)$, where n is large enough for security reasons. One can generate a secret short ring element $g \in R$, in order to generate a principal ideal $\mathcal{I} = \langle g \rangle \subset R$. One should also generate an integer parameter q and another random secret $z \in R/qR$.

  Encoding: One encodes elements of the quotient ring $R/\mathcal{I}$, namely elements of the form $e + \mathcal{I}$ for some e, as follows: a level-i encoding of the coset $e + \mathcal{I}$ is an element of the form $u_i = [c/z^i]_q$, where $c \in e+\mathcal{I}$ is short. Such encodings can be both added and multiplied, as long as the norm of the numerators remain shorter than q; in particular the product of k encodings at level 1 gives an encoding at

level k.

- The second construction [CLT13]:

  One first generates n secret primes p_i and publishes $x_0 = \prod_{i=1}^{n} p_i$ (where n is large enough to ensure correctness and security); one also generates n small secret primes g_i, and a random secret integer z modulo x_0. A level-k encoding of a vector $m = (m_i) \in \mathbb{Z}^n$ is then an integer c such that for all $1 \leq i \leq n$:

  $$c = \frac{r_i \cdot g_i + m_i}{z^k} \pmod{p_i}$$

  for some small random integers r_i; the integer c is therefore defined modulo x_0 by CRT. It is clear that such encodings can be both added and multiplied modulo x_0, as long as the numerators remain smaller than the p_i's. In particular the product of k encodings c_j at level 1 gives an encoding at level k where the corresponding vectors m_j are multiplied component wise.

Comparison between the two schemes: The second scheme was considered (before the aforementioned attack!) to be more practical, and could fit more easily with other existing cryptographic tools since it preserves some of the cryptographic assumptions that didn't hold with the first scheme, e.g. DLIN and subgroup decision. Those assumptions are believed necessary to adapt constructions of modules like adaptively secure functional encryption and NIZK, therefore the second construction seemed more promising for applications than the first one. The multilinear maps topic is currently considered one of the hot cryptographic topics, so one should expect more schemes and cryptanalysis are yet to appear.

3.10 Conclusion

In this chapter, we outlined existing results that we shall be building on in our own work. We defined the *Discrete Logarithm problem* (DLog), and discussed its hardness level in different types of groups. Furthermore, we gave an introduction to the Dlog family, i.e. other hardness assumptions that can be reduced to Dlog. We then went straight to the type of groups that interests us the most, i.e. group of points on a *suitable* elliptic curve. In order to do so, we first gave some useful definitions

about elliptic curves, then defined a very interesting tool in cryptography, i.e. *pairings*, and back again to elliptic curves to define what we meant by suitable elliptic curves, where we defined *pairing-friendly elliptic curves*. Some more assumptions of the Dlog family were also presented, but getting *pairings* involved in them this time. To relate Dlog family members together, we gave a diagram of reductions. Yet another powerful mathematical/cryptographic tool which we rely on in the rest of the thesis, namely, *Zero-Knowledge proofs*, was presented in this chapter. We defined their two main instantiations, i.e. Fiat-Shamir heuristic and Groth-Sahai proofs. At the end, we presented the new promising *multilinear maps* which generalise pairings.

Chapter 4

Decentralised Traceable Attribute Based Signatures

4.1 Introduction

In this chapter, we give a general framework for decentralized traceable attribute based signatures DTABS [EKGK14, EKGK13]. This work is a joint work with Essam Ghadafi (Bristol University) and Dalia Khader (Luxembourg University). In this work, we present the first general framework that covers such important features all together i.e. decentralisation and traceability. Moreover, we will give a generic construction of a DTABS scheme that is secure as long as the underlying cryptographic modules are secure. All the security proofs rely on the security properties that these modules enjoy, but not on any specific scheme. That would allow us to instantiate our DTABS using the most efficient and up-to-date concrete modules. Furthermore, this allows for future improvements whenever more efficient concrete versions of the needed modules appear.

DTABS **main contribution**

- **Formal security model:** We define a formal security model for decentralized traceable attribute-based signatures.
- **Decentralization:** Our focus is on the *more realistic* setting where there are multiple attribute authorities that are responsible for distributing the attributes/credentials to the users, and hence the word *Decentralized*.

- **Traceability:** This is a very important security requirement when it comes to real life applications; *anonymity* is a good feature that ABS usually offer, but it should be controlled in the sense that it cannot be used in a bad way, e.g. framing users, misuse/abuse cases. To solve this problem, we add two entities to the ABS scheme, the first is the *Opener* who will be called to *open* the signatures that are in question in order to repeal the anonymity and therefore tell who has produced these signatures. Moreover, we have a *Judge* who has no secret keys, and yet he is able to tell whether or not the opener has correctly opened the signature, thanks to the NIZK tool.
- **Generic construction for** DTABS**:** We give a security model for DTABS based on the security of the modules needed to realise it, so anyone can actually instantiate the DTABS using different modules as long as they satisfy the security requirements stated in the generic construction.
- **Instantiations:** Moreover, we present two example instantiations of the generic construction and provide the first construction not relying on idealized assumptions. Our constructions meet strong security requirements and permit expressive signing policies.

4.2 Syntax and Security Definitions of DTABS

4.2.1 Syntax of DTABS

The parties involved in a DTABS scheme are: κ attribute authorities each with a unique identity aid and a pair of secret/verification keys $(\mathsf{aask}_{\mathsf{aid}}, \mathsf{aavk}_{\mathsf{aid}})$; a tracing authority T which possesses a secret tracing key tk that can be used to trace the identity of the signer of a given signature; a set of signers each with a unique identity id and a set of attributes $\mathcal{A} \subseteq \mathbb{A}$, where $\mathbb{A}$ is the universe of all possible attributes. An attribute can be uniquely identified by concatenating the identity of the managing attribute authority with the name of the attribute itself. A DTABS scheme is a tuple of polynomial-time algorithms

$$\mathsf{DTABS} = (\mathsf{Setup}, \mathsf{AuthSetup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Trace}, \mathsf{Judge}) \cdot$$

The syntax of the algorithms is defined below; where to aid notation all algorithms

bar Setup and AuthSetup are assumed to take as implicit input the public parameters pp output by algorithm Setup.

- $\mathsf{Setup}(1^\lambda)$ is run by some trusted third party. It takes as input a security parameter 1^λ and outputs public parameters pp and the tracing key tk. We assume that pp contains the attribute universe $\mathbb{A}$.

- $\mathsf{AuthSetup}(\mathsf{pp}, \mathsf{aid})$ used by attribute authority $\mathsf{Auth}_{\mathsf{aid}}$ to generate its key pair $(\mathsf{aask}_{\mathsf{aid}}, \mathsf{aavk}_{\mathsf{aid}})$. The attribute authority publishes its public verification key $\mathsf{aavk}_{\mathsf{aid}}$.

- $\mathsf{KeyGen}(\mathsf{aask}_{\mathsf{aid}}, \mathsf{id}, a)$ takes as input an attribute authority's secret key $\mathsf{aask}_{\mathsf{aid}}$, a signer's identity id and an attribute $a \in \mathbb{A}$ that signer id possesses and generates a secret key $\mathsf{sk}_{\mathsf{id},a}$ for attribute a for the signer. The key $\mathsf{sk}_{\mathsf{id},a}$ is given to id. The attribute authority may locally maintain a list of signers for which it ran the KeyGen algorithm.

- $\mathsf{Sign}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\mathcal{A}}, \{\mathsf{sk}_{\mathsf{id},a}\}_{a\in\mathcal{A}}, m, \Psi)$ Signer id who possesses a set of attributes $\mathcal{A} \subseteq \mathbb{A}$ uses this algorithm to produce a signature on m w.r.t. the signing policy Ψ where $\Psi(\mathcal{A}) = 1$. The algorithm takes as input an ordered list of attribute authorities' verification keys $\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\mathcal{A}}$, an ordered list of attributes' secret keys $\{\mathsf{sk}_{\mathsf{id},a}\}_{a\in\mathcal{A}}$, a message m and a signing predicate Ψ, and outputs a signature σ. Here $\mathsf{aid}(a)$ denotes the identity of the attribute authority managing attribute $a \in \mathbb{A}$.

- $\mathsf{Verify}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi}, m, \sigma, \Psi)$ is a deterministic algorithm which takes as input an ordered list of attribute authorities' verification keys $\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi}$, a message m, a signature σ and a signing predicate Ψ, and outputs 1 if σ is valid on m w.r.t. the signing predicate Ψ or 0 otherwise.

- $\mathsf{Trace}(\mathsf{tk}, m, \sigma, \Psi)$ is a deterministic algorithm which takes as input T's key tk, a message m, a signature σ and a signing predicate Ψ, and outputs the identity id of the signer plus a proof π attesting to this claim. If the algorithm is unable to trace the signature to a signer, it returns the special symbol $\perp$. Note that if the tracing authority additionally gets a read-only access to the local registration tables maintained by the attribute authorities (whose identities can be inferred

from the signing policy Ψ), then the tracing authority could additionally check whether or not id has run the KeyGen algorithm.

- $\mathsf{Judge}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi}, m, \sigma, \Psi, \mathsf{id}, \pi)$ is a deterministic algorithm which takes as input an ordered list of attribute authorities' verification keys $\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi}$, a message m, a signature σ, a signing predicate Ψ, a signer identity id, and a tracing proof π, and outputs 1 if π is a valid proof that id has produced σ or 0 otherwise.

4.2.2 Security Definitions

The security properties required from a DTABS scheme are: correctness, anonymity, full unforgeability, and traceability. In defining those requirements we use a set of experiments in which the adversary has access to a set of oracles. The following global lists are maintained: HSL is a list of honest signers' attributes and has entries of the form (id, a); HAL is a list of honest attribute authorities; BSL is a list of bad signers' attributes whose secret keys have been revealed to the adversary with entries of the form (id, a); BAL is a list of bad attribute authorities whose secret keys have been learned by the adversary; CAL is a list of corrupt attribute authorities whose keys have been chosen by the adversary; SL is a list of signatures obtained from the Sign oracle; CL is a list of challenge signatures obtained from the challenge oracle and is used only in the anonymity game.

The details of the following oracles are given in Fig. 4-1.

- $\mathsf{AddS}(\mathsf{id}, \mathcal{A})$ is used to add honest attributes $\mathcal{A} \subseteq \mathbb{A}$ for signer id. It can be called multiple times to add more attributes.

- $\mathsf{AddA}(\mathsf{aid})$ is used to add an honest attribute authority with identity aid.

- $\mathsf{CrptA}(\mathsf{aid}, \mathsf{vk})$ is used to create a corrupt attribute authority whose secret key is chosen by the adversary.

- $\mathsf{RevealS}(\mathsf{id}, \mathcal{A})$ is used to obtain the secret keys $\{\mathsf{sk}_{\mathsf{id},a}\}_{a\in\mathcal{A}}$ corresponding to the subset of attributes $\mathcal{A} \subseteq \mathbb{A}$ owned by signer id. It can be called multiple times.

- $\mathsf{RevealA}(\mathsf{aid})$ is used to obtain the secret key $\mathsf{aask}_{\mathsf{aid}}$ of the honest attribute authority aid.

$\underline{\mathsf{AddS}(\mathsf{id}, \mathcal{A})}$

- If $\exists a \in \mathcal{A}$ s.t. $(\mathsf{id}, a) \in \mathsf{HSL}$ Then Return $\perp$.
- For each $a \in \mathcal{A}$ Do
  - If $\mathsf{aid}(a) \notin \mathsf{HAL}$ Then
    - If $\mathsf{aid}(a) \in \mathsf{CAL}$ Then Return $\perp$.
    - $\mathsf{AddA}(\mathsf{aid}(a))$.
  - If $\mathsf{aask}_{\mathsf{aid}(a)} = \perp$ Then Return $\perp$.
  - $\mathsf{sk}_{\mathsf{id},a} \leftarrow \mathsf{KeyGen}(\mathsf{aask}_{\mathsf{aid}(a)}, \mathsf{id}, a)$.
- $\mathsf{HSL} := \mathsf{HSL} \cup \{(\mathsf{id}, a)\}_{a \in \mathcal{A}}$.

$\underline{\mathsf{AddA}(\mathsf{aid})}$

- If $\mathsf{aid} \in \mathsf{HAL} \cup \mathsf{CAL}$ Then Return $\perp$.
- $(\mathsf{aask}_{\mathsf{aid}}, \mathsf{aavk}_{\mathsf{aid}}) \leftarrow \mathsf{AuthSetup}(\mathsf{pp}, \mathsf{aid})$.
- $\mathsf{HAL} := \mathsf{HAL} \cup \{\mathsf{aid}\}$.

$\underline{\mathsf{Sign}(\mathsf{id}, \mathcal{A}, m, \Psi)}$

- If $\exists a \in \mathcal{A}$ s.t. $(\mathsf{id}, a) \notin \mathsf{HSL}$ Then Return $\perp$.
- If $\Psi(\mathcal{A}) \neq 1$ or $\exists a \in \mathcal{A}$ s.t. $\mathsf{sk}_{\mathsf{id},a} = \perp$ Then Return $\perp$.
- $\sigma \leftarrow \mathsf{Sign}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a \in \mathcal{A}}, \{\mathsf{sk}_{\mathsf{id},a}\}_{a \in \mathcal{A}}, m, \Psi)$.
- $\mathsf{SL} := \mathsf{SL} \cup \{(\mathsf{id}, \mathcal{A}, m, \sigma, \Psi)\}$.
- Return σ.

$\underline{\mathsf{Trace}(m, \sigma, \Psi)}$

- If $\mathsf{Verify}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a \in \Psi}, m, \sigma, \Psi) = 0$ Then Return $\perp$.
- If $(m, \sigma, \Psi) \in \mathsf{CL}$ Then Return $\perp$.
- Return $\mathsf{Trace}(\mathsf{tk}, m, \sigma, \Psi)$.

$\underline{\mathsf{CH}_b((\mathsf{id}_0, \mathcal{A}_0), (\mathsf{id}_1, \mathcal{A}_1), m, \Psi)}$

- If $\Psi(\mathcal{A}_0) \neq 1$ or $\Psi(\mathcal{A}_1) \neq 1$ Then Return $\perp$.
- For i=0 To 1 Do
  - For each $a \in \mathcal{A}_i$ s.t. $(\mathsf{id}_i, a) \notin \mathsf{HSL}$ DO
    - If $\mathsf{AddS}(\mathsf{id}_i, a) = \perp$ Then Return $\perp$.
  - If $\exists a \in \mathcal{A}_i$ s.t. $\mathsf{sk}_{\mathsf{id}_i,a} = \perp$ Then Return $\perp$.
- $\sigma \leftarrow \mathsf{Sign}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a \in \mathcal{A}_b}, \{\mathsf{sk}_{\mathsf{id}_b,a}\}_{a \in \mathcal{A}_b}, m, \Psi)$.
- $\mathsf{CL} = \mathsf{CL} \cup \{(m, \sigma, \Psi)\}$.
- Return σ.

$\underline{\mathsf{RevealA}(\mathsf{aid})}$

- If $\mathsf{aid} \notin \mathsf{HAL} \setminus (\mathsf{CAL} \cup \mathsf{BAL})$ Then Return $\perp$.
- $\mathsf{BAL} := \mathsf{BAL} \cup \{\mathsf{aid}\}$.
- Return $\mathsf{aask}_{\mathsf{aid}}$.

$\underline{\mathsf{CrptA}(\mathsf{aid}, \mathsf{vk})}$

- If $\mathsf{aid} \in \mathsf{HAL} \cup \mathsf{CAL}$ Then Return $\perp$.
- $\mathsf{CAL} := \mathsf{CAL} \cup \{\mathsf{aid}\}$.

$\underline{\mathsf{RevealS}(\mathsf{id}, \mathcal{A})}$

- If $\exists a \in \mathcal{A}$ s.t. $(\mathsf{id}, a) \notin \mathsf{HSL} \setminus \mathsf{BSL}$ Then Return $\perp$.
- $\mathsf{BSL} := \mathsf{BSL} \cup \{(\mathsf{id}, a)\}_{a \in \mathcal{A}}$.
- Return $\{\mathsf{sk}_{\mathsf{id},a}\}_{a \in \mathcal{A}}$.

Figure 4-1: Oracles used in the security games for DTABS

- $\mathsf{Sign}(\mathsf{id}, \mathcal{A}, m, \Psi)$ is used to obtain a signature σ on message m by signer id using $\{\mathsf{sk}_{\mathsf{id},a}\}_{a \in \mathcal{A}}$ where $\Psi(\mathcal{A}) = 1$.
- $\mathsf{CH}_b((\mathsf{id}_0, \mathcal{A}_0), (\mathsf{id}_1, \mathcal{A}_1), m, \Psi)$ is a left-right oracle for defining anonymity and is only called once. The adversary sends a couple of identities $(\mathsf{id}_0, \mathcal{A}_0)$,

$(\mathsf{id}_1, \mathcal{A}_1)$, a message m and a signing policy Ψ. If $\Psi(\mathcal{A}_0) = \Psi(\mathcal{A}_1) = 1$, the oracle returns a signature on m using $\{\mathsf{sk}_{\mathsf{id}_b,a}\}_{a\in\mathcal{A}_b}$ for $b \leftarrow \{0,1\}$.

- $\mathsf{Trace}(m, \sigma, \Psi)$ allows the adversary to ask for signatures to be traced.

The security requirements are defined by the games in Figs. 4-2 and 4-3.

$\underline{\text{Experiment}\,\mathsf{Exp}^{\text{Trace}}_{\mathsf{DTABS},\mathcal{F}}(\lambda)}$

- $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
- $\mathsf{CAL}, \mathsf{HSL}, \mathsf{HAL}, \mathsf{BSL}, \mathsf{BAL}, \mathsf{SL} := \emptyset$.
- $(m^*, \sigma^*, \Psi^*) \leftarrow \mathcal{F}\,(\mathsf{pp}, \mathsf{tk} : \mathsf{AddS}(\cdot,\cdot), \mathsf{AddA}(\cdot), \mathsf{RevealS}(\cdot,\cdot), \mathsf{Sign}(\cdot,\cdot,\cdot,\cdot))$.
- If $\mathsf{Verify}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi^*}, m^*, \sigma^*, \Psi^*) = 0$ Then Return 0.
- $(\mathsf{id}^*, \pi) \leftarrow \mathsf{Trace}(\mathsf{tk}, m^*, \sigma^*, \Psi^*)$.
- If $\mathsf{id}^* = \perp$ Then Return 1.
- If $\mathsf{Judge}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi^*}, m^*, \sigma^*, \Psi^*, \mathsf{id}^*, \pi) = 0$ Then Return 1.
- If $(\mathsf{id}, \cdot) \in \mathsf{HSL}$ Then Return 0.
- Return 1.

Figure 4-2: Security experiments for DTABS-1

Correctness. This demands that signatures produced by honest signers are accepted by the Verify algorithm and trace to the signer who produced them. Moreover, the Judge algorithm accepts the proof produced by the Trace algorithm. Formally, a DTABS scheme is correct if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{F}$ have a negligible advantage

$$\mathsf{Adv}^{\text{Corr}}_{\mathsf{DTABS},\mathcal{F}}(\lambda) = \Pr[\mathsf{Exp}^{\text{Corr}}_{\mathsf{DTABS},\mathcal{F}}(\lambda) = 1]$$

Anonymity. This requires that a signature reveals neither the identity of the signer nor the set of attributes used in the signing. This is a stronger notion than what is used in other settings, e.g. [Kha07, LK08], which only require that the identity of the signer remains anonymous. Thus, our definition ensures that a signature does not reveal more information other than what can be already inferred from the signing predicate itself.

In the game, the adversary chooses a message, a signing policy and two signers with two, possibly different, sets of attributes with the condition that both sets have to

satisfy the signing policy. The adversary gets a signature by either signer and wins if it correctly guesses the signer.

Our model provides the adversary with strong capabilities, for instance, it can fully corrupt the attribute authorities and can ask for signers' secret keys to be revealed including the two signers it chooses for the challenge (and thus capturing full-key exposure attacks). Note that since the adversary can sign on behalf of any signer, it is redundant to provide the adversary with a sign oracle. The only restriction we impose on the adversary is that it may not query the Trace oracle on the challenge signature.

In CPA-anonymity [BBS04], the adversary is not given access to the Trace oracle. On the contrary, in CCA-anonymity [Men05], the adversary can ask Trace queries at any stage of the game on any signature except the challenge signature. One can also consider a weaker non-adaptive variant of CCA-anonymity where the adversary can only ask Trace queries before it sees the challenge signature.

Also, our definition captures unlinkability because the adversary has access to all signers' secret keys and hence can produce signatures on behalf of any signer.

Formally, a DTABS scheme is anonymous if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{F}$ have a negligible advantage

$$\mathsf{Adv}^{\text{Anon}}_{\mathsf{DTABS},\mathcal{F}}(\lambda) = \left|\Pr[\mathsf{Exp}^{\text{Anon-0}}_{\mathsf{DTABS},\mathcal{F}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\text{Anon-1}}_{\mathsf{DTABS},\mathcal{F}}(\lambda) = 1]\right|$$

Full Unforgeability. This requirement captures unforgeability scenarios where the forgery opens to a particular signer. It ensures that even if signers collude and combine their attributes together, they cannot forge a signature that opens to a signer whose attributes do not satisfy the signing predicate. It also covers non-frameability and ensures that even if signers collude, they cannot frame a user who did not produce the signature.

Unlike the single attribute authority setting, here we allow the adversary to adaptively create corrupt attribute authorities and learn some of the honest authorities' secret keys as long as there is at least a single honest attribute authority managing one of the attributes required for satisfying the policy used in the forgery. In addition, we allow the adversary to fully corrupt the tracing authority.

Our definition is adaptive and allows the adversary to adaptively choose the predicate and the message on which it wants to produce the forgery rather than having to select the predicate at the start of the game. Also, note that we consider the stronger

form of unforgeability, i.e. strong unforgeability where the adversary wins even if it manages to produce a new signature on a message/predicate pair that was queried to the sign oracle. We refer to this stronger definition as *Strong Full Unforgeability* and use the abbreviation SFU for short. The definition can, in a straightforward manner, be adapted to work for the weaker variant used in, e.g. [Men05, MPR11, EHM11], by requiring that the forgery is not on a message/predicate pair that was queried to the sign oracle. For the latter variant which we refer to as *Weak Full Unforgeability* (WFU), we just need to replace the check $\exists(\mathsf{id}^*, \cdot, m^*, \Psi^*, \sigma^*) \in \mathsf{SL}$ by the check $\exists(\mathsf{id}^*, \cdot, m^*, \Psi^*, \cdot) \in \mathsf{SL}$.

Formally, a DTABS scheme is fully unforgeable if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{F}$ have a negligible advantage

$$\mathsf{Adv}^{\text{F-Unforge}}_{\mathsf{DTABS},\mathcal{F}}(\lambda) = \Pr[\mathsf{Exp}^{\text{F-Unforge}}_{\mathsf{DTABS},\mathcal{F}}(\lambda) = 1]$$

Traceability. This requirement ensures that the adversary cannot produce a signature that traces to a signer who did not run the honest KeyGen algorithm. Thus, it covers unforgeability scenarios where the forgery is untraceable. In the game, the adversary is allowed to corrupt the tracing authority and ask for the signing keys of any signer to be revealed. However, unlike in the full unforgeability game, we require that all the attribute authorities are honest as knowing a secret key of any attribute authority makes it easy to create signatures by dummy signers which are thus untraceable.

Formally, a DTABS scheme is traceable if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{F}$ have a negligible advantage

$$\mathsf{Adv}^{\text{Trace}}_{\mathsf{DTABS},\mathcal{F}}(\lambda) = \Pr[\mathsf{Exp}^{\text{Trace}}_{\mathsf{DTABS},\mathcal{F}}(\lambda) = 1]$$

Comparison with Escala et al. Model [EHM11] for the Single Attribute Authority Setting

Specializing our model to the single attribute authority setting, we get a stronger model than the one in [EHM11]. In particular, our model avoids some of the shortcomings inherent to [EHM11] which we now explain. When defining non-frameability in [EHM11], the sign oracle used by [EHM11] does not consider the identity of the signer and hence it does not capture the following scenario: the adversary asks for two different signers id_1 with attributes $\mathcal{A}_1$ and id_2 with attributes $\mathcal{A}_2$ to be added. It then

asks for a signature on the message m w.r.t. the signing policy Ψ by signer id_1 (where $\Psi(\mathcal{A}_1) = 1$), and outputs as its forgery a signature σ^* on the same message m w.r.t. the same signing policy Ψ but the signature opens to id_2 (assume here that $\Psi(\mathcal{A}_2) = 1$).

Therefore, we believe that in this context, where traceability is required, it is important that the identity of the signer is taken into account when answering signing queries. Otherwise, some of the unforgeability scenarios are not captured. This is, of course, different from standard attribute-based signatures where traceability is not required and thus there is no way for the adversary to learn who produced a particular signature.

In addition, our full unforgeability definition protects against a fully corrupt tracing authority which is stronger than the non-frameability definition in [EHM11] which only considers a partially but not fully corrupt tracing authority.

$\underline{\text{Experiment}\mathsf{Exp}^{\text{Corr}}_{\mathsf{DTABS},\mathcal{F}}(\lambda)}$

- $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
- $\mathsf{HSL} := \emptyset$.
- $(\mathsf{id}, \mathcal{A}, m, \Psi) \leftarrow \mathcal{F}(\mathsf{pp} : \mathsf{AddS}(\cdot,\cdot), \mathsf{AddA}(\cdot))$.
- If $\Psi(\mathcal{A}) \neq 1$ or $\mathcal{A} \not\subseteq \mathbb{A}$ Then Return 0.
- If $\exists a \in \mathcal{A}$ s.t. $(\mathsf{id}, a) \notin \mathsf{HSL}$ or $\mathsf{sk}_{\mathsf{id},a} = \perp$ or $\mathsf{aid}(a) \notin \mathsf{HAL}$ Then Return 0.
- $\sigma \leftarrow \mathsf{Sign}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\mathcal{A}}, \{\mathsf{sk}_{\mathsf{id},a}\}_{a\in\mathcal{A}}, m, \Psi)$.
- If $\mathsf{Verify}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi}, m, \sigma, \Psi) = 0$ Then Return 1.
- $(\mathsf{id}', \pi) \leftarrow \mathsf{Trace}(\mathsf{tk}, m, \sigma, \Psi)$.
- If $\mathsf{id}' \neq \mathsf{id}$ Then Return 1.
- If $\mathsf{Judge}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi}, m, \sigma, \Psi, \mathsf{id}, \pi) = 0$ Then Return 1.
- Return 0.

$\underline{\text{Experiment}\mathsf{Exp}^{\text{Anon-}b}_{\mathsf{DTABS},\mathcal{F}}(\lambda)}$

- $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
- $\mathsf{CAL}, \mathsf{HSL}, \mathsf{HAL}, \mathsf{BSL}, \mathsf{BAL}, \mathsf{CL} := \emptyset$.
- $b^* \leftarrow \mathcal{F}(\mathsf{pp} : \mathsf{AddS}(), \mathsf{AddA}(), \mathsf{CrptA}(), \mathsf{RevealS}(), \mathsf{RevealA}(), \mathsf{CH}_b(), \mathsf{Trace}())$.
- Return b^*.

$\underline{\text{Experiment}\mathsf{Exp}^{\text{F-Unforge}}_{\mathsf{DTABS},\mathcal{F}}(\lambda)}$

- $(\mathsf{pp}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
- $\mathsf{CAL}, \mathsf{HSL}, \mathsf{HAL}, \mathsf{BSL}, \mathsf{BAL}, \mathsf{SL} := \emptyset$.
- $(m^*, \sigma^*, \Psi^*, \mathsf{id}^*, \pi^*) \leftarrow \mathcal{F}(\mathsf{pp}, \mathsf{tk} : \mathsf{AddS}(), \mathsf{AddA}(), \mathsf{CrptA}(), \mathsf{RevealS}(), \mathsf{RevealA}(), \mathsf{Sign}())$.
- If $\mathsf{Verify}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi^*}, m^*, \sigma^*, \Psi^*) = 0$ Then Return 0.
- If $\mathsf{Judge}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi^*}, m^*, \sigma^*, \Psi^*, \mathsf{id}^*, \pi^*) = 0$ Then Return 0.
- Let $\mathcal{A}_{\mathsf{id}^*}$ be the set of attributes owned by id^* and managed by dishonest (i.e. $\in \mathsf{CAL} \cup \mathsf{BAL}$) attribute authorities.
- If $\exists \mathcal{A}$ s.t. $\{(\mathsf{id}^*, a)\}_{a\in\mathcal{A}} \subseteq \mathsf{BSL}$ and $\Psi^*(\mathcal{A} \cup \mathcal{A}_{\mathsf{id}^*}) = 1$ Then Return 0.
- If $\exists(\mathsf{id}^*, \cdot, m^*, \sigma^*, \Psi^*) \in \mathsf{SL}$ Then Return 0.
- Return 1.

Figure 4-3: Security experiments for DTABS-2

4.3 Modules needed to Construct DTABS

In this section we present the building blocks that we use in our constructions.

4.3.1 Tagged Signature Scheme

We define here a new variant of a signature scheme which we call a *Tagged Signature* (TS) scheme[1]. A tagged signature scheme for a message space $\mathcal{M}_{\mathsf{TS}}$ and a tag space $\mathcal{T}_{\mathsf{TS}}$ is a tuple of algorithms

$$\mathsf{TS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$$

- $\mathsf{Setup}(1^\lambda)$ this optional algorithm takes as input a security parameter and outputs common public parameters param which is an implicit input to the rest of algorithms.
- $\mathsf{KeyGen}(\{\mathsf{param}|1^\lambda\})$ takes as input either public parameters (if the scheme requires a setup) or just the security parameter (if no setup is required) and outputs a pair of secret/verification keys $(\mathsf{sk}, \mathsf{vk})$.
- $\mathsf{Sign}(\mathsf{sk}, \tau, m)$ takes as input a secret key sk, a tag $\tau \in \mathcal{T}_{\mathsf{TS}}$ and a message $m \in \mathcal{M}_{\mathsf{TS}}$, and outputs a signature σ.
- $\mathsf{Verify}(\mathsf{vk}, \tau, m, \sigma)$ outputs 1 if σ is a signature on τ and m w.r.t. the verification key vk.

The security of a tagged signature scheme is similar to that of a traditional digital signature and consists of correctness and unforgeability:

- **Correctness:** Requires that for all $m \in \mathcal{M}_{\mathsf{TS}}$, $\tau \in \mathcal{T}_{\mathsf{TS}}$ and $(\mathsf{sk}, \mathsf{vk})$ output by KeyGen, we have $\mathsf{Verify}(\mathsf{vk}, \tau, m, \mathsf{Sign}(\mathsf{sk}, \tau, m)) = 1$.
- **(Existential) Unforgeability:** Unforgeability under adaptive chosen-message-tag attack requires that any PPT adversary $\mathcal{F}$ that is given a sign oracle $\mathsf{Sign}(\mathsf{sk}, \cdot, \cdot)$ has a negligible advantage in winning the following game:
  - A key pair $(\mathsf{sk}, \mathsf{vk})$ is generated and vk is sent to $\mathcal{F}$.

[1] A variant of this primitive has been used by [ACHO13] in the context of one-time signatures which they call tagged one-time signatures.

- $\mathcal{F}$ makes a polynomial number of queries to $\mathsf{Sign}(\mathsf{sk}, \cdot, \cdot)$.
- Eventually, $\mathcal{F}$ halts by outputting a tuple (σ^*, τ^*, m^*) and wins if σ^* is valid on (τ^*, m^*) and (τ^*, m^*) was never queried to Sign.

We note here that any signature scheme that can sign a pair of messages can be used as a tagged signature scheme. However, to allow for generality and explicitly distinguish the tag space from the message space (and hence care for the case where they might be distinct), we define this notion. Note that one can always use a collision-resistant hash function to map the tag into the message space.

Defining this as a new notion also serves to simplify the description of our constructions and security proofs. In particular, later in the constructions we need to hide the tag, whereas the message remains public and hence just signing the hash of the combination of the tag and the message using a standard digital signature would be problematic.

Instantiation 1. To construct a tagged signature, we use a variant of the automorphic scheme from [Fuc09] which was given in [Fuc11]. The original scheme given in [Fuc11] was given in the asymmetric bilinear group setting. For simplicity, the variant we give here is in the symmetric setting. The tag space of the tagged signature scheme is Diffie–Hellman tuples $\mathcal{DH}$ where $\mathcal{DH} = \{(G^a, G'^a) \in \mathbb{G}^2 | a \in \mathbb{Z}_p\}$, whereas the message space is $\mathbb{Z}_p$. The scheme is unforgeable under the q-ADHSDH and WFCDH assumptions in the symmetric setting (and the q-ADHSDH and AWFCDH assumptions in the asymmetric setting) (cf. Section 4.3.6). Our tagged signature construction is as follows:

- $\mathsf{TS.Setup}(1^\lambda)$: Let $\mathcal{P} = (\mathbb{G}, \mathbb{G}_T, p, G, e)$ be the description of Type-1 bilinear groups. Choose $F, K, T, G', L \leftarrow \mathbb{G}$ and return $\mathsf{param} = (\mathcal{P}, F, K, T, L, G')$.
- $\mathsf{TS.KeyGen}(\mathsf{param})$: Choose $x \leftarrow \mathbb{Z}_p$ and set $(X, X') = (G^x, G'^x)$. Set $\mathsf{sk} = x$ and $\mathsf{vk} = (X, X')$.
- $\mathsf{TS.Sign}(\mathsf{sk}, (\tau, \tau'), m)$: Reject if $(\tau, \tau') \notin \mathcal{DH}$ (i.e. $e(\tau, G') \neq e(G, \tau')$). Otherwise, choose $u, v \leftarrow \mathbb{Z}_p$ and compute $\sigma = \Big(U = G^u,\ \ U' = G'^u,\ \ V = F^v, V' = G'^v,\ \ W = (K \cdot T^u \cdot \tau \cdot L^m)^{\frac{1}{x+v}}\Big)$.
- $\mathsf{TS.Verify}(\mathsf{vk}, (\tau, \tau'), m, \sigma)$: Check that:
  - $e(U, G') = e(G, U')$,

- $e(V, G') = e(F, V')$,
- $e(W, X' \cdot V') = e(T, U')e(K \cdot \tau \cdot L^m, G')$, and output 1 or 0 accordingly.

Instantiation 2. In [CM14], they provide an efficient randomizable structure preserving signature scheme in type-3 pairings setting that is superior to its counterpart in the type-2 setting as presented in [AGOT14]. The original scheme in [AGOT14] can sign a vector of messages $\{M_1, \cdots, M_n\}$. We will use the case where $n = 2$ in order to sign a couple of messages, i.e. (τ, M).

- $\mathsf{TS.Setup}(1^\lambda)$: Let $\mathcal{P} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, I, e)$ be the description of Type-3 bilinear groups. G and I are fixed generators of $\mathbb{G}_1$ and $\mathbb{G}_2$.
- $\mathsf{TS.KeyGen}(\mathcal{P})$: Choose $u, v, w \leftarrow \mathbb{Z}_p$ and set $(U, V, W) = (G^u, G^v, G^w)$. Set $\mathsf{sk} = (u, v, w)$ and $\mathsf{vk} = (U, V, W)$.
- $\mathsf{TS.Sign}(\mathsf{sk}, (\tau, M) \in \mathbb{G}_2)$: Select $r \leftarrow \mathbb{Z}_p$ and compute $\sigma = \Big(R_1 = G^r,\ \ R_2 = I^r,\ \ S = \tau^u M^v I^{r^2+w}$. The signature is $\sigma = (R_1, R_2, S)$
- $\mathsf{TS.Verify}(\mathsf{vk}, (\tau, M), \sigma)$: Check that;
  - $R_1 \in \mathbb{G}_1$ and $\tau, M, R_2, S \in \mathbb{G}_2$
  - $e(R_1, I) = e(G, R_2)$
  - $e(G, S) = e(U, \tau) \cdot e(V, M) \cdot e(R_1, R_2) \cdot e(W, I)$

  Output 1 or 0 accordingly.

4.3.2 The Full Boneh-Boyen (FBB) Signature Scheme

In [BB04a], the authors gave a signature scheme that is secure under the q-SDH assumption (cf. Section 4.3.6). The signature scheme can be instantiated in both the symmetric and asymmetric bilinear group settings. Let $\mathcal{P} = (\mathbb{G}, \mathbb{G}_T, p, G, e)$ be the description of a bilinear group. The scheme is as follows; where to aid notation all algorithms bar KeyGen are assumed to take as implicit input $\mathcal{P}$:

- $\mathsf{KeyGen}(\mathcal{P})$: Choose $x, y \leftarrow \mathbb{Z}_p$ and set $(X, Y) = (G^x, G^y)$. Set $\mathsf{sk} = (x, y)$ and $\mathsf{vk} = (X, Y)$.

- $\mathsf{Sign}(\mathsf{sk}, m)$: To sign $m \in \mathbb{Z}_p$, choose $r \leftarrow \mathbb{Z}_p$ such that $x + r \cdot y + m \neq 0$ and compute the signature $\sigma = G^{\frac{1}{x+r\cdot y+m}}$.

- $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$: Output 1 if $e(\sigma, X \cdot Y^r \cdot G^m) = e(G, G)$ and 0 otherwise.

4.3.3 Strongly Unforgeable One-Time Signatures

A digital signature scheme is called *one-time signature* if in the unforgeability game, the adversary is restricted to a single signing query. *Strong Unforgeability* as opposed to weak unforgeability requires that the adversary cannot even forge a new signature on a message that she obtained a signature on from the signing oracle. We will instantiate the one-time signature using the Full Boneh-Boyen signature scheme from Section 4.3.2.

4.3.4 Groth-Sahai Proofs.

Groth-Sahai (GS) proofs [GS12] are efficient non-interactive proofs in the Common Reference String (CRS) model. The GS system can be instantiated under different intractability assumptions with the SXDH-based instantiation (see section 4.3.6) being the most efficient [GSW10].

4.3.5 Tag-Based Encryption

A *Tag-based Public-Key Encryption* (TPKE) scheme [MRY04] is similar to a public-key encryption scheme with the only difference being that both Enc and Dec algorithms take as an additional input a tag t.

More formally, a TPKE scheme for a message space $\mathcal{M}_{\mathsf{TPKE}}$ and a tag space $\mathcal{T}_{\mathsf{TPKE}}$ is a tuple of polynomial-time algorithms

$$\mathsf{TPKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{IsValid}),$$

where the algorithms are defined as follows:

- $\mathsf{KeyGen}(1^\lambda)$: Takes a security parameter 1^λ and outputs a public key pk and a secret key sk.

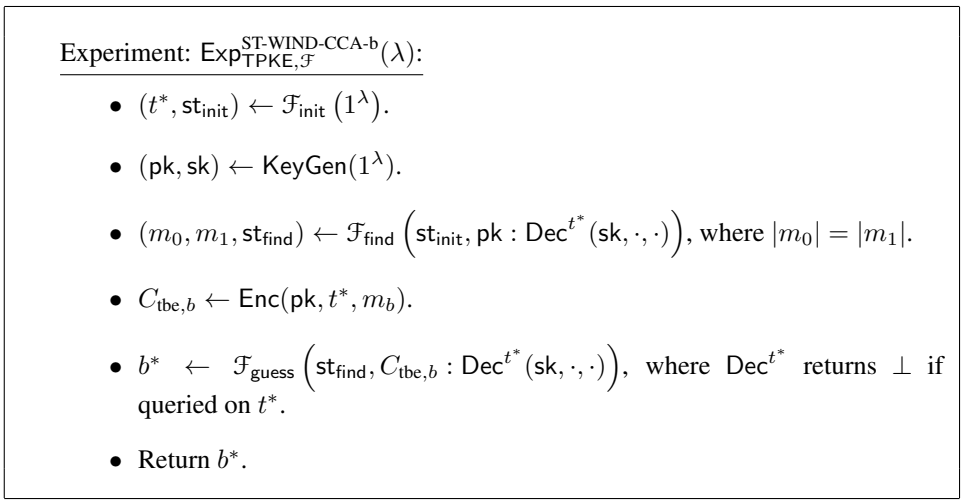


Figure 4-4: ST-WIND-CCA security game for TPKE

- $\mathsf{Enc}(\mathsf{pk}, t, m)$: Takes as input the public key pk, a tag $t \in \mathcal{T}_{\mathsf{TPKE}}$ and a message $m \in \mathcal{M}_{\mathsf{TPKE}}$, and outputs a ciphertext C_{tbe}.
- $\mathsf{Dec}(\mathsf{sk}, t, C_{\text{tbe}})$: Takes as input the secret key sk, a tag $t \in \mathcal{T}_{\mathsf{TPKE}}$ and a ciphertext C_{tbe}, and outputs a message m or the reject symbol $\perp$.
- $\mathsf{IsValid}(\mathsf{pk}, t, C_{\text{tbe}})$: This is an optional algorithm and is used to check whether a ciphertext is valid under the tag t w.r.t. the public key pk. It returns 1 or 0 accordingly.

Besides the usual correctness requirement, we require selective-tag weak indistinguishability against adaptive chosen-ciphertext attacks (ST-WIND-CCA), which is defined by the game in Fig. 4-4.

We say the scheme is ST-WIND-CCA secure if for all $\lambda \in \mathbb{N}$, all polynomial-time adversaries $\mathcal{F}$ have a negligible advantage

$$\mathsf{Adv}^{\text{ST-WIND-CCA}}_{\mathsf{TPKE},\mathcal{F}}(\lambda) = \left|\Pr[\mathsf{Exp}^{\text{ST-WIND-CCA-0}}_{\mathsf{TPKE},\mathcal{F}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\text{ST-WIND-CCA-1}}_{\mathsf{TPKE},\mathcal{F}}(\lambda) = 1]\right|$$

Kiltz [Kil06] showed how to combine a ST-WIND-CCA secure TPKE scheme with a strongly unforgeable one-time signature scheme to obtain an IND-CCA2 secure PKE scheme. In the transformation the one-time signature verification key is used as a tag for the TPKE scheme and then the tag-based ciphertext C_{tbe} is signed with the one-time signature secret signing key.

Instantiation of ST-WIND-CCA Tag-Based Encryption. We use the selective-tag weakly secure CCA tag-based encryption scheme by Kiltz [Kil06] which is secure under the DLIN assumption. The scheme is in Fig. 4-5.

In [Kak10], it was shown that the tag-based scheme in Fig. 4-5 can be translated into both (type-2 & type-3) asymmetric bilinear group settings. The security of the scheme in the type-3 setting relies on a variant of the DLIN assumption called the SDLIN assumption, in which the last element in the input tuple is provided in both groups. However, the security of this variant requires that the message space is polynomial in the security parameter so that we can efficiently search when decrypting.

- $\mathsf{TPKE.KeyGen}(1^\lambda)$
  - $(\mathbb{G}, p) \leftarrow \mathsf{GrpSetup}(1^\lambda)$.
  - $K, L \leftarrow \mathbb{G}$; $f, h \leftarrow \mathbb{Z}_p$.
  - $F = G^f$, $H = G^h$.
  - $\mathsf{pk} = (G, F, H, K, L)$; $\mathsf{sk} = (f, h)$.
- $\mathsf{TPKE.Enc}(\mathsf{pk}, t, M)$
  - $r_1, r_2 \leftarrow \mathbb{Z}_p$.
  - $C_1 = F^{r_1}$; $C_2 = H^{r_2}$; $C_3 = G^{r_1+r_2} \cdot M$.
  - $C_4 = (G^t \cdot K)^{r_1}$; $C_5 = (G^t \cdot L)^{r_2}$.
  - $C_{\text{tbe}} = (C_1, C_2, C_3, C_4, C_5)$.
- $\mathsf{TPKE.Dec}(\mathsf{sk}, t, C_{\text{tbe}})$
  - If $\mathsf{TPKE.IsValid}(\mathsf{pk}, t, C_{\text{tbe}}) = 0$ Then return $\perp$
  - Parse C_{tbe} as $(C_1, C_2, C_3, C_4, C_5)$.
  - $M = C_3 \cdot C_1^{-1/f} C_2^{-1/h}$.
- $\mathsf{TPKE.IsValid}(\mathsf{pk}, t, C_{\text{tbe}})$
  - Parse C_{tbe} as $(C_1, C_2, C_3, C_4, C_5)$.
  - If $e(F, C_4) \neq e(C_1, G^t \cdot K)$ Or $e(H, C_5) \neq e(C_2, G^t \cdot L)$ Then Return 0.
  - Else Return 1.

Figure 4-5: The tag-based encryption by Kiltz [Kil06]

4.3.6 Complexity Assumptions

We will use the following assumptions from the literature:

- DDH For a group $\mathbb{G} = \langle G \rangle$ of a prime order p given $(G, G^a, G^b, C) \in \mathbb{G}^4$ for $a, b \leftarrow \mathbb{Z}_p$, it is hard to decide whether or not $C = G^{ab}$.
- SXDH The Decisional Diffie-Hellman (DDH) assumption holds in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$.
- DLIN [BBS04] In Type-1 groups where $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G} = \langle G \rangle$, given the tuple $(G^a, G^b, G^{ra}, G^{sb}, G^t)$ where $a, b, r, s, t \in \mathbb{Z}_p$ are unknown, it is hard to tell whether $t = r + s$ or t is random.
- q-SDH [BB04a] For a group $\mathbb{G} = \langle G \rangle$ of a prime order p given $(G, G^x, \ldots, G^{x^q})$ for $x \leftarrow \mathbb{Z}_p$, it is hard to output a pair $(c, G^{\frac{1}{x+c}}) \in \mathbb{Z}_p \times \mathbb{G}$ for an arbitrary $c \in \mathbb{Z}_p \backslash \{-x\}$.
- WFCDH [Fuc09]. In symmetric bilinear groups, given $(G, G^a, G^b) \in \mathbb{G}^3$ for $a, b \leftarrow \mathbb{Z}_p$, it is hard to output a tuple $(G^r, G^{ra}, G^{rb}, G^{rab}) \in \mathbb{G}^4$ for an arbitrary $r \in \mathbb{Z}_p$.
- AWFCDH [Fuc09] In asymmetric bilinear groups, given $(G, G^a, \tilde{G}) \in {\mathbb{G}_1}^2 \times \mathbb{G}_2$ for $a \leftarrow \mathbb{Z}_p$, it is hard to output a tuple $(G^b, G^{ab}, \tilde{G}^b, \tilde{G}^{ab}) \in {\mathbb{G}_1}^2 \times {\mathbb{G}_2}^2$ for an arbitrary $b \in \mathbb{Z}_p$.
- q-ADHSDH [Fuc09] In asymmetric bilinear groups [1], given $(G, F, K, G^x, \tilde{G}, \tilde{G}^x) \in \mathbb{G}_1^4 \times \mathbb{G}_2^2$ for $x \leftarrow \mathbb{Z}_p$, and $q-1$ tuples $(W_i = (K \cdot G^{u_i})^{\frac{1}{x+v_i}}, U_i = G^{u_i}, \tilde{U}_i = \tilde{G}^{u_i}, V_i = F^{v_i}, \tilde{V}_i = \tilde{G}^{v_i})_{i=1}^{q-1}$ for $u_i, v_i \leftarrow \mathbb{Z}_p$, it is hard to output a new tuple $(W^*, U^*, \tilde{U}^*, V^*, \tilde{V}^*)$ of this form.

4.3.7 Span Programs

A *span program* [KW93] is defined as follows:

Definition 9. *Given a monotone boolean function $\Phi : \{0,1\}^n \rightarrow \{0,1\}$, a $l \times t$ matrix M with entries in a field $\mathbb{F}$, and a labelling function $\rho : [l] \rightarrow [n]$ that associates M's*

[1] This can also be instantiated in symmetric groups. See [Fuc09].

rows to Φ's input variables. We say that M is a monotone span program for ϕ over a field $\mathbb{F}$ if for every $(x_1, \ldots, x_n) \in \{0,1\}^n$, we have the following:

$$[\Phi(x_1, \ldots, x_n) = 1] \Leftrightarrow [\exists \vec{v} \in \mathbb{F}^{1\times t} : \vec{v} \cdot M = (1, 0, \cdots, 0) \\ \wedge (\forall i : x_{\rho(i)} = 0 \Rightarrow v_i = 0)]$$

In other words, let $I = \{i : x_{\rho(i)} = 1\}$, and M_I the corresponding sub-matrix of M,

$$\Phi(x_1, \cdots, x_n) = 1 \Leftrightarrow \textit{the rows of } M_I \textit{ span the vector } (1, 0, \cdots, 0).$$

Example 5. *[Bei11] Consider the following monotone span program $(\mathbb{F}_{17}, M, \rho)$ where;*

$$M = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{pmatrix}$$

Let $\rho(1) = \rho(2) = x_2, \rho(3) = x_1$, and $\rho(4) = x_3$. Consider the sets $I_1 = \{1, 2, 3\}$ and $I_2 = \{3, 4\}$ which correspond to $\{x_1, x_2\}$ and $\{x_1, x_3\}$ respectively. In this case,

$$M_{I_1} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{pmatrix} \quad \textit{and} \quad M_{I_2} = \begin{pmatrix} 1 & 3 & 9 \\ 1 & 4 & 16 \end{pmatrix}$$

As M_{I_1} has full rank, the rows of M_{I_1} span $(1, 0, 0)$, i.e., there exist a vector $\vec{v'} = (3, 14, 1)$ s.t. $\vec{v'} \cdot M_{I_1} = (1, 0, 0)$, in $\mathbb{F}_{17}$. It is easy to see that for $\vec{v} = (3, 14, 1, 0)$, we have $\vec{v} \cdot M = (1, 0, 0, 0)$ in $\mathbb{F}_{17}$ as well. Hence, the span program accepts $\{x_1, x_2\}$. On the other hand, the rows of M_{I_2} do not span the vector $(1, 0, 0)$ and therefore the span program does not accept $\{x_1, x_3\}$. Note that the minimal authorized sets in the access structure accepted by M are $\{x_1, x_2\}$ and $\{x_2, x_3\}$.

See [Bei96, Bei11] for more details on span programs and their relation to *Linear Secret Sharing Schemes* (LSSS) [KGH83].

4.4 First Generic Construction

In this section, we present our generic construction for decentralized traceable attribute-based signatures.

Overview of the construction. The tools we use in our generic construction are two NIZK proof systems NIZK_1 and NIZK_2, an IND-CCA2 secure public-key encryption scheme PKE, an existentially unforgeable tagged signature scheme TS, and an existentially unforgeable digital signature scheme DS with a message space $\mathcal{M}_{\mathsf{DS}}$. In addition, we need a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \to \mathcal{M}_{\mathsf{DS}}$.

We require that the NIZK_1 proof system, which will be used in the signing, is simulation-sound [Sah99] and a proof of knowledge [SP92]. In fact, it is sufficient for it to be only one-time simulation-sound (see section 3.8.2). On the contrary, it suffices that NIZK_2 is a zero-knowledge proof system, i.e. we require neither simulation-soundness nor knowledge extractability from NIZK_2.

The Setup algorithm generates two separate common reference strings crs_1 and crs_2 for the NIZK systems NIZK_1 and NIZK_2, respectively. It also generates a key pair $(\mathsf{tvk}, \mathsf{tsk})$ for the digital signature scheme DS, and an encryption/decryption key pair $(\mathsf{epk}, \mathsf{esk})$ for the encryption scheme PKE. The public parameters of the system is set to $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{tvk}, \mathsf{epk}, \mathbb{A}, \mathcal{H})$, where $\mathbb{A}$ is the universe of attributes. The tracing authority's key is set to $\mathsf{tk} = \mathsf{esk}$.

When a new attribute authority joins the system, it creates a secret/verification key pair $(\mathsf{aask}_{\mathsf{aid}}, \mathsf{aavk}_{\mathsf{aid}})$ for the tagged signature scheme TS. To generate a signing key for attribute $a \in \mathbb{A}$ for signer id, the managing attribute authority signs the signer identity id (used as tag) along with the attribute a using her secret tagged signature signing key. The resulting signature is used as the secret key for that attribute by signer id.

To sign a message m w.r.t. a signing policy Ψ, the signer first encrypts her identity id using the encryption scheme PKE (and some randomness μ) to obtain a ciphertext C. She then computes, using the NIZK system NIZK_1, a proof π that she encrypted her identity correctly and that she either has a digital signature on the hash of the combination of the signing predicate, the message and the ciphertext containing her identity, i.e. $\mathcal{H}(\Psi, m, C)$, that verifies w.r.t. the verification key tvk or that she owns enough attributes to satisfy the original signing predicate Ψ in the form of tagged signatures on her identity and the attributes. For ease of composition and following

- $\underline{\mathsf{Setup}(1^\lambda)}$
  - $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$ and $\mathsf{crs}_2 \leftarrow \mathsf{NIZK}_2.\mathsf{Setup}(1^\lambda)$.
  - $(\mathsf{tvk}, \mathsf{tsk}) \leftarrow \mathsf{DS.KeyGen}(1^\lambda)$ and $(\mathsf{epk}, \mathsf{esk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda; \rho)$.
  - Choose a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathcal{M}_{\mathsf{DS}}$.
  - Let $\mathsf{tk} = \mathsf{esk}$ and $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{tvk}, \mathsf{epk}, \mathbb{A}, \mathcal{H})$, where $\mathbb{A}$ is the attribute universe.
  - Return pp.
- $\underline{\mathsf{AuthSetup}(\mathsf{pp}, \mathsf{aid})}$
  - $(\mathsf{aavk}_{\mathsf{aid}}, \mathsf{aask}_{\mathsf{aid}}) \leftarrow \mathsf{TS.KeyGen}(1^\lambda)$.
  - Return $(\mathsf{aavk}_{\mathsf{aid}}, \mathsf{aask}_{\mathsf{aid}})$.
- $\underline{\mathsf{KeyGen}(\mathsf{aask}_{\mathsf{aid}(a)}, \mathsf{id}, a)}$
  - $\mathsf{sk}_{\mathsf{id},a} \leftarrow \mathsf{TS.Sign}(\mathsf{aask}_{\mathsf{aid}(a)}, \mathsf{id}, a)$.
  - Return $\mathsf{sk}_{\mathsf{id},a}$.
- $\underline{\mathsf{Sign}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\mathcal{A}}, \{\mathsf{sk}_{\mathsf{id},a}\}_{a\in\mathcal{A}}, m, \Psi)}$
  - Return $\perp$ if $\Psi(\mathcal{A}) = 0$.
  - $C \leftarrow \mathsf{PKE.Enc}(\mathsf{epk}, \mathsf{id}; \mu)$.
  - Let $\hat{\Psi} = \Psi \vee a_{\Psi,m,C}$ and $\mathbf{Z} \in \mathbb{Z}_p^{|\hat{\Psi}|,\beta}$ be the span program for $\hat{\Psi}$.
  - Let $\vec{a} = \{a_i\}_{i=1}^{|\hat{\Psi}|}$ denote the attributes appearing in $\hat{\Psi}$.
  - $\pi \leftarrow \mathsf{NIZK}_1.\mathsf{Prove}(\mathsf{crs}_1, \{\mathsf{id}, \mu, \vec{s}, \{\sigma_{a_i}\}_{i=1}^{|\hat{\Psi}|}\} : (C, \{\mathsf{aavk}_{\mathsf{aid}(a_i)}\}_{i=1}^{|\hat{\Psi}|-1} \cup \mathsf{tvk}, \vec{a}) \in \mathcal{L}_1)$.
  - Return $\sigma = (\pi, C)$.

Figure 4-6: The generic construction for DTABS-1

[MPR11], we refer to $\mathcal{H}(\Psi, m, C)$ as pseudo-attributes and denote them by $a_{\Psi,m,C}$. Note here that including the ciphertext as part of the encoding of the pseudo-attribute does not affect the signature size.

The extended predicate $\hat{\Psi}$ is proved via a span program (see section 4.3.7) represented by the matrix $\mathbf{Z}$: the signer proves that she knows a secret vector $\vec{s} \in \mathbb{Z}_p^{|\hat{\Psi}|}$ s.t. $\vec{s}\mathbf{Z} = [1, 0, \ldots, 0]$. She also needs to show that she possesses a valid (tagged) signature on each attribute in the signing predicate for which the corresponding element in $\vec{s}$ is non-zero or a valid signature that verifies w.r.t tvk in the case of a pseudo-attribute. For attributes appearing in the policy that the signer does not own, she chooses random signatures.

Note that the hiding properties of the NIZK_1 system ensure that the proof π does

- $\underline{\mathsf{Verify}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi}, m, \sigma, \Psi)}$
  - Return $\mathsf{NIZK}_1.\mathsf{Verify}(\mathsf{crs}_1, \pi)$.
- $\underline{\mathsf{Trace}(\mathsf{tk}, m, \sigma, \Psi)}$
  - Return $(\perp, \perp)$ if $\mathsf{Verify}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi}, m, \sigma, \Psi) = 0$.
  - $\mathsf{id} \leftarrow \mathsf{PKE.Dec}(\mathsf{tk}, C)$.
  - $\pi_{\mathsf{Trace}} \leftarrow \mathsf{NIZK}_2.\mathsf{Prove}(\mathsf{crs}_2, \{\mathsf{tk}\} : (C, \mathsf{epk}, \mathsf{id}) \in \mathcal{L}_2)$.
  - Return $(\mathsf{id}, \pi_{\mathsf{Trace}})$.
- $\underline{\mathsf{Judge}(\mathsf{id}, \pi_{\mathsf{Trace}}, m, \sigma, \Psi)}$
  - If $(\mathsf{id}, \pi_{\mathsf{Trace}}) = (\perp, \perp)$ Then Return $\mathsf{Verify}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi}, m, \sigma, \Psi) = 0$.
  - Return $\mathsf{NIZK}_2.\mathsf{Verify}(\mathsf{crs}_2, \pi_{\mathsf{Trace}})$.

Figure 4-7: The generic construction for DTABS-2

not reveal how the modified predicate $\hat{\Psi}$ was satisfied, i.e. whether the signer has a special signature on the pseudo-attribute or she owns enough attributes to satisfy the original predicate Ψ. The signature is then set to $\sigma = (\pi, C)$. To verify the signature, one just needs to verify the proof π.

The modified OR predicate $\hat{\Psi}$ serves to bind the signature to the message and the signing predicate. The secret signing key tsk for the digital signature scheme DS is only used as a trapdoor in the security proofs, and thus is not given to any authority. It allows its holder to simulate signatures and sign on behalf of any signer without knowing their secret keys by simply encrypting their identity and producing a signature on the pseudo-attribute associated with the message and the signing predicate. Note that even in the unlikely case that any of the pseudo-attributes happened to collide with a real attribute, this is not a problem since signatures associated with pseudo-attributes must verify w.r.t. tvk which is different from all attribute authorities' keys.

To trace a signature, the tracing authority just decrypts the ciphertext C to recover the signer's identity. It then produces a proof π_{Trace} using the NIZK system NIZK_2 to prove that the decryption was done correctly. To verify the tracing correctness, the judge just needs to verify the validity of the NIZK proof π_{Trace}.

The details of the construction are in Figs. 4-6 and 4-7, whereas the languages associated with the NIZK proofs used in the construction are as follows, where for

clarity we underline the elements of the witness:

$$\mathcal{L}_1 : \Big\{ \big((C, \vec{\mathsf{vk}} = \{\mathsf{aavk}_{\mathsf{aid}(a_i)}\}_{i=1}^{|\hat{\Psi}|-1} \cup \mathsf{tvk}, \vec{a} = \{a_i\}_{i=1}^{|\hat{\Psi}|}), (\mathsf{id}, \mu, \vec{s}, \vec{\sigma} = \{\sigma_{a_i}\}_{i=1}^{|\hat{\Psi}|})\big) : \\ \Big(\underline{\vec{s}}\mathbf{Z} = [1, 0, \ldots, 0] \bigwedge_{i=1}^{|\hat{\Psi}|-1} \text{ if } \underline{s_i} \neq 0 \Rightarrow \mathsf{TS.Verify}(\mathsf{vk}_i, \underline{\mathsf{id}}, a_i, \underline{\sigma_{a_i}}) = 1 \\ \wedge \text{ if } \underline{s_{|\hat{\Psi}|}} \neq 0 \Rightarrow \mathsf{DS.Verify}(\mathsf{tvk}, a_{\Psi,m,C}, \underline{\sigma_{a_{|\hat{\Psi}|}}}) = 1\Big) \\ \wedge \ \mathsf{PKE.Enc}(\mathsf{epk}, \underline{\mathsf{id}}; \underline{\mu}) = C\Big\}.$$

The witness consists of a signer identity id, the randomness μ used in encrypting id, a vector $\vec{s} \in \mathbb{Z}_p^{|\hat{\Psi}|}$, and signatures $\{\sigma_{a_i}\}_{i=1}^{|\hat{\Psi}|}$ s.t. the span program $\mathbf{Z}$ verifies w.r.t. to $\vec{s}$ and for every non-zero element s_i for $i \in \{1, \ldots |\vec{s}| - 1\}$, the tagged signature σ_{a_i} on id (as a tag) and the attribute a_i (as a message) verifies w.r.t. the corresponding attribute authority verification key, and if $s_{|\vec{s}|} \neq 0$, the signature $\sigma_{|\hat{\Psi}|}$, i.e. the one on the special pseudo-attribute verifies w.r.t. the verification key tvk.

$$\mathcal{L}_2 : \Big\{ \big((C, \mathsf{epk}, \mathsf{id}), (\mathsf{tk}, \rho)\big) : \mathsf{PKE.KeyGen}(1^\lambda; \underline{\rho}) = (\mathsf{epk}, \underline{\mathsf{tk}}) \\ \wedge \ \mathsf{PKE.Dec}(\underline{\mathsf{tk}}, C) = \mathsf{id}\Big\}.$$

The witness consists of the tracing key, i.e. the decryption key for PKE, and the randomness ρ (if any) used in the key generation of PKE s.t. the encryption/decryption key pair is correct and the ciphertext C decrypts to id.

Note that if we encrypted the whole witness of π (rather than just the signer identity) then we could drop the requirement for NIZK_1 to be a proof of knowledge. The reason why we cannot afford to do this is two-fold: first, since the decryption key is used as a tracing key and signers do not have their own personal key pairs, this would mean that a dishonest tracing authority will be able to forge on behalf of an honest signer once it has opened a signature by them. Second, since in both the full unforgeability and traceability experiments, the adversary has access to the tracing key, it would mean that we can no longer sign using pseudo-attributes since the adversary will be able to learn what witness we used in producing a signature.

Also, note that for the construction to satisfy the stronger variant of full unforgeability (i.e. SFU) rather than WFU, NIZK_1 must additionally be strongly non-malleable in the sense that it is infeasible for the adversary to even output a new proof for a state-

ment that it received a proof for. In particular, as noted by [Gro06] if the proof system is simulation-sound extractable [Gro06] then it is non-malleable.

Theorem 3. *The construction in Figs. 4-6 and 4-7 is a secure decentralized traceable attribute-based signature if the building blocks are secure w.r.t. their security requirements.*

The proof will follow in section 4.8.

4.5 Second Generic Construction

In order to get more efficient constructions in the standard model, we slightly deviate from the generic framework by dropping the requirement that NIZK_1 is simulation-sound. In particular, in our instantiations we will use the Groth-Sahai proof system (which is the only efficient non-interactive proof system not relying on random oracles) to instantiate both NIZK_1 and NIZK_2 systems. Note that Groth-Sahai proofs are malleable and therefore not simulation-sound. Although there exist transformations which make Groth-Sahai proofs simulation-sound, e.g. [Gro06], unfortunately, all those transformations degrade the efficiency of the proofs. Also, note that the fact that one cannot efficiently extract exponents from Groth-Sahai proofs is not a problem in our case as we never need to be able to efficiently extract the exponent components of the witness.

We will start by describing the idea of the construction generically and then present the exact instantiations later. To eliminate the need for NIZK_1 to be simulation-sound, we apply a trick similar to that used by Groth in [Gro07] where we sign the final signature with a strongly unforgeable one-time signature scheme OTS. We require that OTS is strongly existentially unforgeable against adaptive chosen-message attack. We use the corresponding one-time verification key as a tag for a selective-tag weakly IND-CCA (i.e. ST-WIND-CCA secure) tag-based encryption scheme TPKE with which we encrypt the user's identity id. The rest of the tools are the same as in the generic construction 4.4.

To map the one-time signature verification key into the tag space of the tag-based encryption, we require another collision-resistant hash function, $\hat{\mathcal{H}} : \{0,1\}^* \rightarrow \mathcal{T}_{\mathsf{TPKE}}$. In order to further bind the signature to the one-time signature verification key (i.e. the tag used for the ciphertext), we sign the one-time signature verification key as a part of

the pseudo-attribute, i.e. the pseudo-attribute now is $(\Psi, m, C_{\text{tbe}}, \hat{\mathcal{H}}(\mathsf{otsvk}))$, which we will denote by $a_{\Psi,m,C_{\text{tbe}},\hat{\mathcal{H}}(\mathsf{otsvk})}$. The one-time signature serves to prevent the adversary from transforming a signature that it received into another valid signature as it now must be able to forge a one-time signature in order to succeed.

The general idea of this construction is given in Figs. 4-8 and 4-9, whereas the languages associated with the NIZK proofs used in the construction are as follows, where again the elements of the witness are underlined:

- Language $\mathcal{L}_1'$ is defined as

$$\begin{aligned}\mathcal{L}_1' : \Big\{ \big((\hat{\mathcal{H}}(\mathsf{otsvk}), C_{\text{tbe}}, \vec{\mathsf{vk}} = \{\mathsf{aavk}_{\mathsf{aid}(a_i)}\}_{i=1}^{|\hat{\Psi}|-1} \cup \mathsf{tvk}, \vec{a} = \{a_i\}_{i=1}^{|\hat{\Psi}|}),\\ (\mathsf{id}, \mu, \vec{s}, \vec{\sigma} = \{\sigma_{a_i}\}_{i=1}^{|\hat{\Psi}|})\big) :\\ \Big(\underline{\vec{s}}\mathbf{Z} = [1, 0, \ldots, 0] \bigwedge_{i=1}^{|\hat{\Psi}|-1} \text{ if } \underline{s_i} \neq 0 \Rightarrow \mathsf{TS.Verify}(\mathsf{vk}_i, \underline{\mathsf{id}}, a_i, \underline{\sigma_{a_i}}) = 1\\ \wedge \ \text{ if } \underline{s_{|\hat{\Psi}|}} \neq 0 \Rightarrow \mathsf{DS.Verify}(\mathsf{tvk}, a_{\Psi,m,C_{\text{tbe}},\hat{\mathcal{H}}(\mathsf{otsvk})}, \underline{\sigma_{a_{|\hat{\Psi}|}}}) = 1\Big)\\ \wedge \ \mathsf{TPKE.Enc}(\mathsf{epk}, \hat{\mathcal{H}}(\mathsf{otsvk}), \underline{\mathsf{id}}; \underline{\mu}) = C_{\text{tbe}} \Big\}.\end{aligned}$$

- Language $\mathcal{L}_2'$ is defined as

$$\begin{aligned}\mathcal{L}_2' : \Big\{ \big((\hat{\mathcal{H}}(\mathsf{otsvk}), C_{\text{tbe}}, \mathsf{epk}, \mathsf{id}), (\mathsf{tk}, \rho)\big) : \ \mathsf{TPKE.KeyGen}(1^\lambda; \underline{\rho}) = (\mathsf{epk}, \underline{\mathsf{tk}})\\ \wedge \ \mathsf{TPKE.Dec}(\underline{\mathsf{tk}}, \hat{\mathcal{H}}(\mathsf{otsvk}), C_{\text{tbe}}) = \mathsf{id} \Big\}.\end{aligned}$$

Theorem 4. *The construction in Figs. 4-8 and 4-9 is a secure decentralized traceable attribute-based signature if the building blocks are secure w.r.t. their security requirements.*

The proof will follow in section 4.8.

- $\underline{\mathsf{Setup}(1^\lambda)}$
  - $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$ and $(\mathsf{crs}_2, \mathsf{xk}_2) \leftarrow \mathsf{NIZK}_2.\mathsf{Setup}(1^\lambda)$.
  - $(\mathsf{tvk}, \mathsf{tsk}) \leftarrow \mathsf{DS.KeyGen}(1^\lambda)$ and $(\mathsf{epk}, \mathsf{esk}) \leftarrow \mathsf{TPKE.KeyGen}(1^\lambda; \rho)$.
  - Choose collision-resistant hash functions $\mathcal{H} : \{0,1\}^* \to \mathcal{M}_{\mathsf{DS}}$ and $\hat{\mathcal{H}} : \{0,1\}^* \to \mathcal{T}_{\mathsf{TPKE}}$.
  - Let $\mathsf{tk} = \mathsf{esk}$ and $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{tvk}, \mathsf{epk}, \mathbb{A}, \mathcal{H}, \hat{\mathcal{H}})$, where $\mathbb{A}$ is the attribute universe.
  - Return pp.
- $\underline{\mathsf{AuthSetup}(\mathsf{pp}, \mathsf{aid})}$
  - $(\mathsf{aavk}_{\mathsf{aid}}, \mathsf{aask}_{\mathsf{aid}}) \leftarrow \mathsf{TS.KeyGen}(1^\lambda)$.
  - Return $(\mathsf{aavk}_{\mathsf{aid}}, \mathsf{aask}_{\mathsf{aid}})$.
- $\underline{\mathsf{KeyGen}(\mathsf{aask}_{\mathsf{aid}(a)}, \mathsf{id}, a)}$
  - $\mathsf{sk}_{\mathsf{id},a} \leftarrow \mathsf{TS.Sign}(\mathsf{aask}_{\mathsf{aid}(a)}, \mathsf{id}, a)$.
  - Return $\mathsf{sk}_{\mathsf{id},a}$.
- $\underline{\mathsf{Sign}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\mathcal{A}}, \{\mathsf{sk}_{\mathsf{id},a}\}_{a\in\mathcal{A}}, m, \Psi)}$
  - Return $\perp$ if $\Psi(\mathcal{A}) = 0$.
  - $(\mathsf{otsvk}, \mathsf{otssk}) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$.
  - $C_{\text{tbe}} \leftarrow \mathsf{TPKE.Enc}(\mathsf{epk}, \hat{\mathcal{H}}(\mathsf{otsvk}), \mathsf{id}; \mu)$.
  - Let $\hat{\Psi} = \Psi \vee a_{\Psi, m, C_{\text{tbe}}, \hat{\mathcal{H}}(\mathsf{otsvk})}$ and $\mathbf{Z} \in \mathbb{Z}_p^{|\hat{\Psi}|, \beta}$ be the span program for $\hat{\Psi}$.
  - Let $\vec{a} = \{a_i\}_{i=1}^{|\hat{\Psi}|}$ denote the attributes appearing in $\hat{\Psi}$.
  - $\pi \leftarrow \mathsf{NIZK}_1.\mathsf{Prove}(\mathsf{crs}_1, \{\mathsf{id}, \mu, \vec{s}, \{\sigma_{a_i}\}_{i=1}^{|\hat{\Psi}|}\} : (\hat{\mathcal{H}}(\mathsf{otsvk}), C_{\text{tbe}}, \{\mathsf{aavk}_{\mathsf{aid}(a_i)}\}_{i=1}^{|\hat{\Psi}|-1} \cup \mathsf{tvk}, \vec{a}) \in \mathcal{L}_1')$.
  - $\sigma_{\text{ots}} \leftarrow \mathsf{OTS.Sign}(\mathsf{otssk}, (\pi, C_{\text{tbe}}, \mathsf{otsvk}))$.
  - Return $\sigma = (\sigma_{\text{ots}}, \pi, C_{\text{tbe}}, \mathsf{otsvk})$.
- $\underline{\mathsf{Verify}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi}, m, \sigma, \Psi)}$
  - Parse σ as $(\sigma_{\text{ots}}, \pi, C_{\text{tbe}}, \mathsf{otsvk})$.
  - Return 1 if all the following verify; otherwise, return 0:
    - $\mathsf{OTS.Verify}(\mathsf{otsvk}, (\pi, C_{\text{tbe}}, \mathsf{otsvk}), \sigma_{\text{ots}}) = 1$.
    - $\mathsf{NIZK}_1.\mathsf{Verify}(\mathsf{crs}_1, \pi) = 1$.
    - $\mathsf{TPKE.IsValid}(\mathsf{epk}, \hat{\mathcal{H}}(\mathsf{otsvk}), C_{\text{tbe}}) = 1$.

Figure 4-8: Details of the second construction-1

- $\underline{\mathsf{Trace}(\mathsf{tk}, m, \sigma, \Psi)}$
  - Return $(\perp, \perp)$ if $\mathsf{Verify}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi}, m, \sigma, \Psi) = 0$.
  - $\mathsf{id} \leftarrow \mathsf{TPKE.Dec}(\mathsf{tk}, \hat{\mathcal{H}}(\mathsf{otsvk}), C_{\mathsf{tbe}})$.
  - $\pi_{\mathsf{Trace}} \leftarrow \mathsf{NIZK}_2.\mathsf{Prove}(\mathsf{crs}_2, \{\mathsf{tk}, \rho\} : (\hat{\mathcal{H}}(\mathsf{otsvk}), C_{\mathsf{tbe}}, \mathsf{epk}, \mathsf{id}) \in \mathcal{L}_2')$.
  - Return $(\mathsf{id}, \pi_{\mathsf{Trace}})$.
- $\underline{\mathsf{Judge}(\mathsf{id}, \pi_{\mathsf{Trace}}, m, \sigma, \Psi)}$
  - If $(\mathsf{id}, \pi_{\mathsf{Trace}}) = (\perp, \perp)$ Then Return $\mathsf{Verify}(\{\mathsf{aavk}_{\mathsf{aid}(a)}\}_{a\in\Psi}, m, \sigma, \Psi) = 0$.
  - Return $\mathsf{NIZK}_2.\mathsf{Verify}(\mathsf{crs}_2, \pi_{\mathsf{Trace}})$.

Figure 4-9: Details of the second construction-2

4.6 An Instantiation in Symmetric Groups

We use the instantiation of the tagged signature scheme from Section 4.3.1 and instantiate the digital signature DS used for pseudo-attributes with the full Boneh-Boyen signature scheme (cf. Section 4.3.2) both in the symmetric setting. Thus, we assume a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p$. Note that we need not hide the integer component r of the full Boneh-Boyen signature when proving π as such a signature can only be generated by the simulator running the security game and hence r does not reveal any information about the attributes involved or the identity of the signer. In other words, in both the real signature and the simulated signature cases, r is chosen uniformly at random.

We use the selective-tag weakly IND-CCA tag-based encryption scheme by Kiltz [Kil06] as illustrated in Fig. 4-5 to instantiate TPKE and instantiate the one-time signature with the full Boneh-Boyen signature in the symmetric setting.

We now give the specific details of the proofs involved. Let $\mathbf{Z} \in \mathbb{Z}_p^{|\hat{\Psi}|,\beta}$ be the span program for $\hat{\Psi} = \Psi \vee a_{\Psi,m,C_{\mathsf{tbe}},\hat{\mathcal{H}}(\mathsf{otsvk})}$. To sign, we need the following proofs:

- To prove that $\vec{s}\mathbf{Z} = [1, 0, \ldots, 0]$, we need to prove the following linear equations:

$$\sum_{i=1}^{|\hat{\Psi}|}(\underline{s_i} Z_{i,1}) = 1 \qquad \sum_{i=1}^{|\hat{\Psi}|}(\underline{s_i} Z_{i,j}) = 0, \text{ for } j = 2, \ldots, \beta \tag{4.1}$$

  To prove that if $\underline{s_i} \neq 0 \Rightarrow \mathsf{TS.Verify}(\mathsf{vk}_i, \underline{\mathsf{id}}, a_i, \underline{\sigma_{a_i}}) = 1$, one needs to raise each pairing involved in the signature verification equations to s_i. This will ensure that if $s_i = 1$ then the only way for the equations to verify is by having a valid signature on id and a_i. On the other hand, when the user does not own attribute a_i, i.e. does not have a valid signature σ_{a_i}, then $s_i = 0$ and the equations will verify since each pairing will evaluate to 1. Based on the observation that computing the components U, U', V, V' of the tagged signature does not require knowledge of the secret signing key and hence even when the user does not have a valid signature on a_i can still choose random components U, U', V, V' of the correct form to satisfy the first two verification equations of the tagged signature. Thus, it is sufficient to use s_i only in the last equation of the tagged signature verification equations. This reduces the number of additional GS commitments and equations required and hence improves the efficiency.

For each of the first $|\hat{\Psi}| - 1$ rows in $\mathbf{Z}$, we prove:

$$\underline{\bar{T}_i} = T^{\underline{s_i}} \qquad \underline{\bar{G}'_i} = G'^{\underline{s_i}} \qquad \underline{\bar{W}_i} = \underline{W_i}^{\underline{s_i}} \tag{4.2}$$

$$e(\underline{U_i}, G'_i) = e(G_i, \underline{U'_i}) \qquad e(\underline{V_i}, G'_i) = e(F_i, \underline{V'_i}) \tag{4.3}$$

$$e(\underline{\bar{W}_i}, X' \cdot \underline{V'_i}) = e(\underline{\bar{T}_i}, \underline{U'_i}) e(K \cdot \underline{\mathsf{id}} \cdot L^{a_i}, \underline{\bar{G}'_i}) \tag{4.4}$$

For the last row in $\mathbf{Z}$, i.e. the pseudo-attribute, the proofs required are:

$$\underline{\bar{\sigma}} = \underline{\sigma}^{\underline{s_{|\hat{\Psi}|}}} \qquad \underline{\bar{G}} = G^{\underline{s_{|\hat{\Psi}|}}} \qquad e(\underline{\bar{\sigma}}, X \cdot Y^r \cdot G^{\mathcal{H}(\Psi, m, C_{\text{tbe}}, \hat{\mathcal{H}}(\mathsf{otsvk}))}) e(\underline{\bar{G}}, G) = 1 \tag{4.5}$$

- To prove that $\mathsf{TPKE.Enc}(\mathsf{epk}, \hat{\mathcal{H}}(\mathsf{otsvk}), \underline{\mathsf{id}}; (\underline{r_1}, \underline{r_2})) = C_{\text{tbe}}$, the signer needs to prove that she computed the ciphertext $(C_1, C_2, C_3, C_4, C_5) = \left(F^{r_1}, H^{r_2}, G^{r_1+r_2} \cdot \mathsf{id}, (G^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot K)^{r_1}, (G^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot L)^{r_2}\right)$ correctly. Since the validity of the ciphertext is publicly verifiable, and for the sake of efficiency, it is sufficient to provide proofs that C_1, C_2 and C_3 were computed correctly and the rest can be verified by checking that $e(F, C_4) = e(C_1, G^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot K)$ and $e(H, C_5) = e(C_2, G^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot L)$. Thus, proving this clause requires proving the 3 following multi-scalar multiplication equations, where the first two are linear, whereas the last equation is quadratic

$$C_1 = F^{\underline{r_1}} \qquad C_2 = H^{\underline{r_2}} \qquad C_3 = G^{\underline{r_1}} \cdot G^{\underline{r_2}} \cdot \underline{\mathsf{id}} \tag{4.6}$$

  Note that we do not need to efficiently extract the exponents r_1 and r_2 from the proofs. Also, note that the equations are simulatable and thus yield zero-knowledge proofs.

- Finally, the signer needs to prove that her identity is a Diffie–Hellman tuple satisfying $e(\underline{\mathsf{id}}, G') = e(G, \underline{\mathsf{id}'})$.

The total size of the Groth-Sahai proofs used is $\mathbb{Z}_p^{2\cdot\beta} + \mathbb{G}^{42\cdot|\hat{\Psi}|-5}$. The proofs require $9 \cdot |\hat{\Psi}| - 1$ GS commitments each of size $\mathbb{G}^3$. The size of the tag-based ciphertext is $\mathbb{G}^5$, whereas the size of the one-time signature including the verification key is $\mathbb{G}^3 + \mathbb{Z}_p$. Thus, the total size of the signature is $\mathbb{Z}_p^{2\cdot\beta+1} + \mathbb{G}^{69\cdot|\hat{\Psi}|}$. An important observation

is that the verification of the signature could be made more efficient by using batch verification techniques for Groth-Sahai proofs [GSW09, BFI+10].

Tracing. Computing the proof π_{Trace} requires proving the following equations

$$G^{\underline{f}} = F \qquad G^{\underline{h}} = H \qquad C_3 \cdot C_1^{-1/\underline{f}} \cdot C_2^{-1/\underline{h}} = \mathsf{id} \tag{4.7}$$

Those are 3 linear MSME proofs and immediately yield zero-knowledge. The total size of π_{Trace} is $\mathbb{G}^{12}$.

The proof for the following Theorem follows from that of Theorem 4.

Theorem 5. *The construction is secure if the assumptions DLIN, q-SDH, q-ADHSDH, and WFCDH hold.*

4.7 Instantiations in Asymmetric Groups

4.7.1 Instantiation 1

To improve efficiency, here we translate the above instantiation into the asymmetric setting (i.e. Type-3 bilinear groups) where we use the more efficient SXDH-based instantiation of Groth-Sahai proofs. We use the asymmetric variants of all the building blocks used in the symmetric instantiation. Note that the security of the asymmetric instantiation of the tag-based encryption scheme from [Kak10] which we use here is based on the SDLIN assumption [Kak10] (a variant of the DLIN assumption in which the last element in the input tuple is provided in both groups) requires that the message space of the encryption scheme (i.e. the number of signers' identities to be encrypted) is polynomial in the security parameter so that we can efficiently search when decrypting. Thus, this instantiation only works when traceability is defined w.r.t. registered users in the system which is polynomial in the security parameter.

For clarity, in the following description we will also accent exponents with˜ when they are to be committed to in group $\mathbb{G}_2$. Let $\mathbf{Z} \in \mathbb{Z}_p^{|\hat{\Psi}|,\beta}$ be the span program for $\hat{\Psi} = \Psi \vee a_{\Psi,m,C_{\mathsf{tbe}},\hat{\mathcal{H}}(\mathsf{otsvk})}$. To sign, we need the following proofs:

- Commit to the vector $\vec{s}$ in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$ and prove that the values are equal which involves $|\hat{\Psi}|$ proofs for QE of the form $s_i - \tilde{s_i} = 0$.

- To prove that $\vec{s}\mathbf{Z} = [1, 0, \ldots, 0]$, we need to prove the following linear equations:

$$\sum_{i=1}^{|\hat{\Psi}|}(\underline{s_i} Z_{i,1}) = 1 \qquad \sum_{i=1}^{|\hat{\Psi}|}(\underline{s_i} Z_{i,j}) = 0, \text{ for } j = 2, \ldots, \beta \tag{4.8}$$

To prove that if $\underline{s_i} \neq 0 \Rightarrow \mathsf{TS.Verify}(\mathsf{vk}_i, \underline{\mathsf{id}}, a_i, \underline{\sigma_{a_i}}) = 1$, for each of the first $|\hat{\Psi}| - 1$ rows in $\mathbf{Z}$, we prove:

$$\underline{\bar{T}_i} = T^{\underline{s_i}} \qquad \underline{\bar{\tilde{G}}_i} = \tilde{G}^{\underline{s_i}} \qquad \underline{\bar{W}_i} = \underline{W_i}^{\underline{\tilde{s}_i}} \tag{4.9}$$

$$e(\underline{U_i}, \tilde{G}_i) = e(G_i, \underline{\tilde{U}_i}) \qquad e(\underline{V_i}, \tilde{G}_i) = e(F_i, \underline{\tilde{V}_i}) \tag{4.10}$$

$$e(\underline{\bar{W}_i}, \tilde{X} \cdot \underline{\tilde{V}_i}) = e(\underline{\bar{T}_i}, \underline{\tilde{U}_i}) e(K \cdot \underline{\mathsf{id}} \cdot L^{a_i}, \underline{\bar{\tilde{G}}_i}) \tag{4.11}$$

For the last row in $\mathbf{Z}$, i.e. the pseudo-attribute, the proofs required are:

$$\underline{\bar{\sigma}} = \underline{\sigma}^{\underline{\tilde{s}_{|\hat{\Psi}|}}} \qquad \underline{\bar{G}} = G^{\underline{\tilde{s}_{|\hat{\Psi}|}}} \qquad e(\underline{\bar{\sigma}}, \tilde{X} \cdot \tilde{Y}^r \cdot \tilde{G}^{\mathcal{H}(\Psi, m, C_{\text{tbe}}, \hat{\mathcal{H}}(\mathsf{otsvk}))}) e(\underline{\bar{G}}, \tilde{G}) = 1 \tag{4.12}$$

- To prove that $\mathsf{TPKE.Enc}(\mathsf{epk}, \hat{\mathcal{H}}(\mathsf{otsvk}), \underline{\mathsf{id}}; (\underline{r_1}, \underline{r_2})) = C_{\text{tbe}}$, the signer needs to prove that she computed the ciphertext $(C_1, C_2, C_3, \tilde{C}_4, \tilde{C}_5) = (F^{r_1}, H^{r_2}, G^{r_1+r_2} \cdot \mathsf{id}, (\tilde{G}^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot \tilde{K})^{r_1}, (\tilde{G}^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot \tilde{L})^{r_2})$ correctly. Since the validity of the ciphertext is publicly verifiable, and for the sake of efficiency, it is sufficient to provide proofs that C_1, C_2 and C_3 were computed correctly and the rest can be verified by checking that $e(F, \tilde{C}_4) = e(C_1, \tilde{G}^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot \tilde{K})$ and $e(H, \tilde{C}_5) = e(C_2, \tilde{G}^{\hat{\mathcal{H}}(\mathsf{otsvk})} \cdot \tilde{L})$. Thus, proving this clause requires proving the 3 following multi-scalar multiplication equations, where the first two are linear, whereas the last equation is quadratic

$$C_1 = F^{\underline{\tilde{r_1}}} \qquad C_2 = H^{\underline{\tilde{r_2}}} \qquad C_3 = G^{\underline{\tilde{r_1}}} \cdot G^{\underline{\tilde{r_2}}} \cdot \underline{\mathsf{id}} \tag{4.13}$$

Note that we do not need to efficiently extract the exponents $\tilde{r_1}$ and $\tilde{r_2}$ from the proofs. Also, note that the equations are simultable and thus yield zero-knowledge proofs.

- Finally, the signer needs to prove that her identity is a Diffie–Hellman tuple satisfying $e(\underline{\mathsf{id}}, \tilde{G}) = e(G, \underline{\tilde{\mathsf{id}}})$.

The total size of the signature in this setting is $\mathbb{G}_1^{34\cdot|\hat{\Psi}|-2} + \mathbb{G}_2^{32\cdot|\hat{\Psi}|-4} + \mathbb{Z}_p^{\beta+1}$. Again, the verification of the signature could be made more efficient by using batch verification techniques for Groth-Sahai proofs [GSW09, BFI$^+$10].

Tracing. Computing the proof π_{Trace} requires proving the following equations

$$G^{\underline{\tilde{f}}} = F \qquad G^{\underline{\tilde{h}}} = H \qquad C_3 \cdot C_1^{-1/\underline{\tilde{f}}} \cdot C_2^{-1/\underline{\tilde{h}}} = \mathsf{id} \tag{4.14}$$

Those are 3 linear MSME proofs and immediately yield zero-knowledge. The total size of π_{Trace} is $\mathbb{G}_1^3 \times \mathbb{G}_2^4$.

The proof for the following Theorem follows from that of Theorem 4.

Theorem 6. *The construction is secure for a polynomial (in λ) signer identity space if the assumptions SDLIN in $\mathbb{G}_1$, q-SDH, q-ADHSDH, AWFCDH, and SXDH hold.*

We end by noting (similarly to [Kak10]) that by translating the instantiation into the Type-2 setting, we can eliminate the requirement for the signer identity space (i.e. the message space of the TPKE scheme) to be polynomial. In this setting, we can use the instantiation of Groth-Sahai proofs based on DDH in $\mathbb{G}_1$ and DLIN in $\mathbb{G}_2$ as in [GSW10].

4.7.2 Instantiation 2

In this instantiation, we replace the tagged based signature used in the first instantiation by a randomizable structure preserving signature [CM14] (see 4.3.1). The rest of the tools stay the same hence we are not going to repeat the common parts. The schemes in [CM14, AGOT14] appeared recently and thus replacing the tag signature schemes used in [EKGK14, Gha14] by this new scheme would greatly minimize the number of terms in the final DTABS signature, which also shows the importance of building cryptographic schemes generically whenever this is possible. As already mentioned in [CM14], the signature components R_1 and R_2 can be made public, so only the second PPE equation of the signature verification needs to be transformed when used with GS proofs. We need to raise each equation of the $\hat{\Psi} - 1$ signatures to the corresponding

Span program vector component i.e. s_i. Below are the details:

$$\underline{\bar{G}_i} = G^{\underline{\tilde{s}_i}} \qquad \underline{\bar{U}_i} = U^{\underline{\tilde{s}_i}} \qquad \underline{\bar{V}_i} = V^{\underline{\tilde{s}_i}} \qquad \underline{\bar{R}_{1i}} = R_1^{\underline{\tilde{s}_i}} \qquad \underline{\bar{W}_i} = W^{\underline{\tilde{s}_i}}$$

$$e(\underline{\bar{G}_i}, S) = e(\underline{\bar{U}_i}, \underline{\tau}) \cdot e(\underline{\bar{V}_i}, M) \cdot e(\underline{\bar{R}_{1i}}, R_2) \cdot e(\underline{\bar{W}_i}, I)$$

One can easily see that the factor $|\hat{\Psi}|$ in the signature size comes from the tagged signature, and therefore using this efficient scheme as a tagged signature in our DTABS should yield a smaller signature. We have commitments of size $\mathbb{G}_1^{6|\hat{\Psi}|-6} + \mathbb{G}_2^{6|\hat{\Psi}|-5}$. We have one quadratic PPE of size $\mathbb{G}_1^{4|\hat{\Psi}|-4} + \mathbb{G}_2^{4|\hat{\Psi}|-4}$, we also have 5 MSE each of size $\mathbb{G}_1^{2|\hat{\Psi}|-2} + \mathbb{G}_2^{4|\hat{\Psi}|-4}$. Adding these group elements up to the rest of the signature as presented in the first asymmetric instantiation would give a final signature of size $\mathbb{G}_1^{24|\hat{\Psi}|-3} + \mathbb{G}_2^{26|\hat{\Psi}|} + \mathbb{Z}_p^{\beta+1}$

4.8 Proofs

In this section, we will present the security proofs of the first generic construction of DTABS, i.e. Theorem 3. Those of Theorem 4 concerning the second generic construction of DTABS are slightly different, thus we present them in appendix B.

Correctness of the construction follows from that of the underlying building blocks.

Lemma 1. *If the NIZK proof system* NIZK_1 *is simulation-sound and zero-knowledge,* NIZK_2 *is zero-knowledge, the encryption scheme* PKE *is IND-CCA2 secure, and the hash function* $\mathcal{H}$ *is collision-resistant then the generic construction is fully anonymous (against full-key exposure).*

Proof. We show that if there exists an adversary $\mathcal{B}$ which breaks the anonymity of the construction, we can construct adversaries $\mathcal{F}_1$ against the NIZK property of the proof system NIZK_1, $\mathcal{F}_2$ against the NIZK property of the proof system NIZK_2, $\mathcal{F}_{3,d}$ for $d \in \{0,1\}$ against the IND-CCA2 security of the encryption scheme PKE, $\mathcal{F}_4$ against the simulation-soundness of NIZK_1, and $\mathcal{F}_5$ against the collision-resistance of the hash function $\mathcal{H}$ such that

$$\begin{aligned}\mathsf{Adv}^{\text{Anon}}_{\mathsf{DTABS},\mathcal{B}}(\lambda) \leq & 2 \cdot (\mathsf{Adv}^{\text{NIZK}}_{\mathsf{NIZK}_1,\mathcal{F}_1}(\lambda) + \mathsf{Adv}^{\text{NIZK}}_{\mathsf{NIZK}_2,\mathcal{F}_2}(\lambda)) + \mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{PKE},\mathcal{F}_{3,0}}(\lambda) \\ & + \mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{PKE},\mathcal{F}_{3,1}}(\lambda) + \mathsf{Adv}^{\text{SS}}_{\mathsf{NIZK}_1,\mathcal{F}_4}(\lambda) + \mathsf{Adv}^{\text{Coll}}_{\mathcal{H},\mathcal{F}_5}(\lambda).\end{aligned}$$

By the collision-resistance of the $\mathcal{H}$, $\mathcal{B}$ has a negligible advantage in finding pairs $(\Psi^*, m^*) \neq (\Psi, m)$ such that $\mathcal{H}(\Psi^*, m^*, C) = \mathcal{H}(\Psi, m, C)$ for some ciphertext C. If this is not the case, we can use $\mathcal{B}$ to construct an adversary $\mathcal{F}_5$ which breaks the collision-resistance of the hash function $\mathcal{H}$. Thus, from now on we assume that there are no hash collisions.

- **Adversary $\mathcal{F}_1$:** Adversary $\mathcal{F}_1$ runs the Setup algorithm normally and chooses all the keys itself. The only two differences here is that the CRS crs_1 used for NIZK_1 is obtained from $\mathcal{F}_1$'s environment, whereas crs_2 used for NIZK_2 is chosen by $\mathcal{F}_1$ by running $(\mathsf{crs}_2, \mathsf{xk}_2) \leftarrow \mathsf{NIZK}_2.\mathsf{SimSetup}(1^\lambda)$. $\mathcal{F}_1$ forwards $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathsf{tvk}, \mathbb{A}, \mathcal{H})$ to $\mathcal{B}$.

  When asked AddA queries, $\mathcal{F}_1$ chooses the secret/verification keys for the authorities itself. Thus, $\mathcal{F}_1$ can answer any AddS queries itself.

  To answer Trace queries, $\mathcal{F}_1$ just decrypts the ciphertext within the signature, and simulates the proof π_{Trace}.

  When asked a $\mathsf{CH}_b((\mathsf{id}_0, \mathcal{A}_0), (\mathsf{id}_1, \mathcal{A}_1), m, \Psi)$ query, $\mathcal{F}_1$ randomly chooses $b \leftarrow \{0, 1\}$ and generates the signature by signer id_b and then forwards the details of the witness for the signature to its environment which responsds with a proof π which $\mathcal{F}_1$ needs to tell if it is a real proof or a simulated one. $\mathcal{F}_1$ constructs the rest of the signature and forwards it to $\mathcal{B}$. If $\mathcal{B}$'s output is equal to the bit b, then $\mathcal{F}_1$ outputs "real", otherwise outputs "simulated".

- **Adversary $\mathcal{F}_2$:** Adversary $\mathcal{F}_2$ runs the Setup algorithm normally and chooses all the keys itself. The only difference here is the the CRS crs_2 used for NIZK_2 is obtained from $\mathcal{F}_2$'s environment.

  Adversary $\mathcal{F}_2$ forwards $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathsf{tvk}, \mathbb{A}, \mathcal{H})$ to $\mathcal{B}$.

  When asked AddA queries, $\mathcal{F}_2$ chooses the secret/verification keys for the authorities itself. Thus, $\mathcal{F}_2$ can answer any AddS queries itself.

  To answer Trace queries, $\mathcal{F}_2$ just decrypts the ciphertext within the signature, and the proof π_{Trace} is obtained from $\mathcal{F}_2$'s Prove oracle. To answer the challenge query $\mathsf{CH}_b((\mathsf{id}_0, \mathcal{A}_0), (\mathsf{id}_1, \mathcal{A}_1), m, \Psi)$, $\mathcal{F}_2$ randomly chooses $b \leftarrow \{0, 1\}$ and generates the signature by signer id_b and constructs the proof π by itself. It then forwards the challenge signature to $\mathcal{B}$. At the end, the output of $\mathcal{F}_2$ is 1 if $\mathcal{B}$ guesses the bit b correctly, and 0 otherwise.

- **Adversaries $\mathcal{F}_{3,0}$ and $\mathcal{F}_{3,1}$:** The details of adversaries $\mathcal{F}_{3,0}$ and $\mathcal{F}_{3,1}$ against the IND-CCA security of PKE are identical except for the difference in answering $\mathcal{B}$'s challenge query.

  Adversary $\mathcal{F}_{3,d}$ gets epk from its IND-CCA game (therefore it does not know the corresponding decryption key esk) and chooses crs_1 for NIZK_1 and crs_2 for NIZK_2 as simulated strings. In its IND-CCA game, $\mathcal{F}_{3,d}$ has access to a decryption oracle Dec.

  Adversary $\mathcal{F}_{3,d}$ starts $\mathcal{B}$ with input $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathsf{tvk}, \mathbb{A}, \mathcal{H})$. When asked AddA queries, $\mathcal{F}_{3,d}$ chooses the secret/verification keys for the authorities itself. Thus, $\mathcal{F}_{3,d}$ can answer any $\mathsf{AddS/Sign}$ queries itself. To answer Trace queries, $\mathcal{F}_{3,d}$ sends the ciphertext C used in the input signature to its decryption oracle and then simulates the proof π_{Trace} (since it does not know the tracing key).

  When asked a $\mathsf{CH}_b((\mathsf{id}_0, \mathcal{A}_0), (\mathsf{id}_1, \mathcal{A}_1), m, \Psi)$ query, $\mathcal{F}_{3,d}$ randomly chooses $d \leftarrow \{0, 1\}$ and sets $p_d = \mathsf{id}_d$ and $p_{1-d} = 0^{|\mathsf{id}_d|}$. It then uses (p_0, p_1) as the challenge pair in its IND-CCA game. When it receives the challenge ciphertext C, $\mathcal{F}_{3,d}$ constructs the rest of the challenge signature by simulating the proof π.

  The rest of $\mathcal{B}$'s queries are answered normally as in Fig. 4-1.

  If in the game, $\mathcal{B}$ queries the Trace oracle on a signature (π', C), i.e. one that re-uses the challenge ciphertext C but the associated proof is different from that used in the challenge signature, then $\mathcal{F}_{3,d}$ outputs its guess d; otherwise, it outputs whatever $\mathcal{B}$ outputs.

- **Adversary $\mathcal{F}_4$:** Adversary $\mathcal{F}_4$ runs the Setup algorithm normally and chooses all the keys itself. The only two differences here is that the CRS crs_1 used for NIZK_1 is obtained from $\mathcal{F}_1$'s simulation-soundness environment and crs_2 for NIZK_2 is chosen by running $(\mathsf{crs}_2, \mathsf{xk}_2) \leftarrow \mathsf{NIZK}_2.\mathsf{SimSetup}(1^\lambda)$. $\mathcal{F}_4$ forwards $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathsf{tvk}, \mathbb{A}, \mathcal{H})$ to $\mathcal{B}$.

  When asked AddA queries, $\mathcal{F}_4$ chooses the secret/verification keys for the authorities itself. Thus, $\mathcal{F}_4$ can answer any AddS queries itself.

  To answer Trace queries, $\mathcal{F}_4$ just decrypts the ciphertext within the signature, and simulates the proof π_{Trace}.

  When asked a $\mathsf{CH}_b((\mathsf{id}_0, \mathcal{A}_0), (\mathsf{id}_1, \mathcal{A}_1), m, \Psi)$ query, $\mathcal{F}_4$ computes C as the encryption of the string of zeros of length equal to the bit length of the signer

identity space and uses the simulated proof it gets from its game to construct the challenge signature that it forwards to $\mathcal{B}$.

If during the game, $\mathcal{B}$ managed to query the Trace oracle on a signature $\sigma' = (\pi', C)$, i.e. one that re-uses the same challenge ciphertext but associated with a different proof π', $\mathcal{F}_4$ outputs the statement with which the proof π' is associated along with π' as its answer in its simulation-soundness game. Otherwise, it aborts.

□

Lemma 2. *The generic construction is fully unforgeable if the NIZK proof systems* NIZK_1 *and* NIZK_2 *are sound, the hash function* $\mathcal{H}$ *(used in encoding pseudo-attributes) is collision-resistant, and the digital signature scheme* DS *and the tagged signature scheme* TS *are both existentially unforgeable.*

Proof. Since the NIZK proof systems NIZK_1 and NIZK_2 are sound, the adversary has a negligible advantage in breaking full unforgeability by faking proofs for a false statement. Also, by the security of the hash function $\mathcal{H}$, the adversary has a negligible probability in finding collisions between the encodings of different pairs of message/signing predicate. Thus, we proceed to show that if there exists an adversary that wins the full unforgeability game then we can construct adversaries $\mathcal{F}_1$ against the unforgeability of the tagged signature scheme TS, and adversary $\mathcal{F}_2$ against the unforgeability of the digital signature scheme DS such that

$$\mathsf{Adv}^{\text{F-Unforge}}_{\mathsf{DTABS},\mathcal{B}}(\lambda) \leq \kappa(\lambda) \cdot \mathsf{Adv}^{\text{Unfor}}_{\mathsf{TS},\mathcal{F}_1}(\lambda) + \mathsf{Adv}^{\text{Unfor}}_{\mathsf{DS},\mathcal{F}_2}(\lambda),$$

where $\kappa(\lambda)$ is a polynomial in λ representing an upper bound on the number of honest attribute authorities $\mathcal{B}$ is allowed to use in the game.

- **Adversay $\mathcal{F}_1$:** Adversary $\mathcal{F}_1$ gets the tagged signature scheme's verification key vk from its game and has access to an oracle Sign that it uses to obtain tagged signatures that verify w.r.t. vk on messages and tags (i.e. identities and attributes) of its choice. Adversary $\mathcal{F}_1$ starts by running $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$, $(\mathsf{crs}_2, \mathsf{xk}_2) \leftarrow \mathsf{NIZK}_2.\mathsf{Setup}(1^\lambda)$ and creating $(\mathsf{tsk}, \mathsf{tvk})$ honestly. It also creates the key pair $(\mathsf{esk}, \mathsf{epk})$ for the encryption scheme. It then forwards $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathsf{tvk}, \mathbb{A}, \mathcal{H})$ and $\mathsf{tk} = \mathsf{esk}$ to $\mathcal{B}$.

Adversary $\mathcal{F}_1$ randomly chooses $i \leftarrow \{1, \ldots, \kappa(\lambda)\}$ and guesses that $\mathcal{B}$'s forgery will involve forging an attribute managed by the attribute authority i. When asked AddA queries, for all authorities $j \neq i$, $\mathcal{F}_1$ chooses the secret/verification keys for the authority itself. For authority i, it sets its verification key to vk it got from its game (and thus it does not know the corresponding secret key). If in the game, $\mathcal{B}$ issues RevealA query on authority i, $\mathcal{F}_1$ aborts the game.

Whenever $\mathcal{B}$ asks AddS queries, if the user has attributes managed by authority i, it forwards such a query to its Sign oracle; otherwise, it answers the query itself by using the authorities' secret keys available to it. When asked for Sign queries on $(\mathsf{id}, \mathcal{A}, m, \Psi)$, $\mathcal{F}_1$ answers the query by first encrypting the identity id and producing a signature on the pseudo-attribute that verifies w.r.t tvk. Note that $\mathcal{F}_1$ knows tsk and hence it can produce such a signature. By the witness-indistinguishability of NIZK_1 (implied by the zero-knowledge property) $\mathcal{B}$ cannot tell how the modified predicate was satisfied and hence cannot distinguish this signature from a real signature where the actual attributes of the user are used. The rest of $\mathcal{B}$'s queries are answered normally as in Fig. 4-1.

Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{F}_1$ uses the NIZK_1's extraction key xk_1 to extract the witness and returns the tagged signature on the identity and the attribute if $\mathcal{B}$'s forgery involved forging a tagged signature. Otherwise, it aborts (i.e. if the forgery was by forging a pseudo-attribute). If the signature does not involve forged attributes managed by authority i that $\mathcal{F}_1$ has guessed, it aborts. The probability that $\mathcal{F}_1$ guesses the correct authority is $\frac{1}{\kappa(\lambda)}$.

- **Adversary $\mathcal{F}_2$:** By the collision-resistant property of the hash function $\mathcal{H}$, the adversary cannot find collisions between different message/predicate pairs and hence we ignore this case.

  Adversary $\mathcal{F}_2$ gets tvk from its game and has access to an oracle Sign that it uses to obtain digital signatures that verify w.r.t. tvk on messages of its choice. It runs $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$, $(\mathsf{crs}_2, \mathsf{xk}_2) \leftarrow \mathsf{NIZK}_2.\mathsf{Setup}(1^\lambda)$. It also creates the key pair $(\mathsf{esk}, \mathsf{epk})$ for the encryption scheme. It then forwards $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathsf{tvk}, \mathbb{A}, \mathcal{H})$ and $\mathsf{tk} = \mathsf{esk}$ to $\mathcal{B}$.

  When asked for AddA queries, $\mathcal{F}_2$ creates the authority keys itself. Whenever $\mathcal{B}$ asks AddS queries, $\mathcal{F}_2$ uses the corresponding authorities' secret keys $\mathsf{aask}_{\mathsf{aid}(a)}$ to create the key for the signer. When asked for Sign queries on $(\mathsf{id}, \mathcal{A}, m, \Psi)$,

$\mathcal{F}_2$ first encrypts id and queries its Sign oracle to obtain a signature on the corresponding pseudo-attribute matching (Ψ, m) and then constructs the rest of the signature by generating the proof π. Again, by the witness-indistinguishability of NIZK_1 (implied by the zero-knowledge property) $\mathcal{B}$ cannot tell which witness was used in the proof. The rest of $\mathcal{B}$'s queries are answered normally as in Fig. 4-1.

Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{F}_2$ uses NIZK_1's extraction key xk_1 to extract the witness and returns the signature on the pseudo-attribute $a_{\hat{\Psi}^*,m^*}$ if the forgery was done by forging a signature on a pseudo-attribute; otherwise, it aborts.

□

Lemma 3. *The generic construction is traceable if the NIZK proof systems* NIZK_1 *is sound, and the digital signature scheme* DS *and the tagged signature scheme* TS *are both existentially unforgeable.*

Proof. The details are very similar to that of full unforgeability. The difference between the full unforgeability proof and the traceability proof is that here the adversary is not allowed to corrupt or learn the secret key of any attribute authority; otherwise, it is easy to create untraceable signature.

Since the NIZK proof systems NIZK_1 is sound, the adversary has a negligible advantage in succeeding by faking proofs for false statements. Thus, we proceed to show that if there exists an adversary that wins the traceability game then we can construct adversaries $\mathcal{F}_1$ against the unforgeability of the tagged signature scheme TS, and adversary $\mathcal{F}_2$ against the unforgeability of the digital signature scheme DS such that

$$\mathsf{Adv}^{\text{Trace}}_{\mathsf{DTABS},\mathcal{B}}(\lambda) \leq \kappa(\lambda) \cdot \mathsf{Adv}^{\text{Unfor}}_{\mathsf{TS},\mathcal{F}_1}(\lambda) + \mathsf{Adv}^{\text{Unfor}}_{\mathsf{DS},\mathcal{F}_2}(\lambda),$$

where $\kappa(\lambda)$ is a polynomial in λ representing an upper bound on the number of honest attribute authorities $\mathcal{B}$ is allowed to use in the game.

- **Adversary $\mathcal{F}_1$:** Adversary $\mathcal{F}_1$ gets the tagged signature scheme's verification key vk from its game and has access to an oracle Sign that it uses to obtain tagged signatures that verify w.r.t. vk on messages and tags (i.e. identities and attributes) of its choice. Adversary $\mathcal{F}_1$ starts by running $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$,

$(\mathsf{crs}_2, \mathsf{xk}_2) \leftarrow \mathsf{NIZK}_2.\mathsf{Setup}(1^\lambda)$ and creating $(\mathsf{tsk}, \mathsf{tvk})$ honestly. It also creates the key pair $(\mathsf{esk}, \mathsf{epk})$ for the encryption scheme. It then forwards $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathsf{tvk}, \mathbb{A}, \mathcal{H})$ and $\mathsf{tk} = \mathsf{esk}$ to $\mathcal{B}$.

Adversary $\mathcal{F}_1$ randomly chooses $i \leftarrow \{1, \ldots, \kappa(\lambda)\}$ and guesses that $\mathcal{B}$'s untraceable signature involves the attribute authority i. When asked AddA queries, for all authorities $j \neq i$, $\mathcal{F}_1$ chooses the secret/verification keys for the authority itself. For authority i, it sets its verification key to vk it got from its game (and thus it does not know the corresponding secret key).

Whenever $\mathcal{B}$ asks AddS queries, if the user has attributes managed by authority i, it forwards such a query to its Sign oracle; otherwise, it answers the query itself by using the authorities' secret keys available to it. When asked for Sign queries on $(\mathsf{id}, \mathcal{A}, m, \Psi)$, $\mathcal{F}_1$ answers the query by producing a signature on the pseudo-attribute that verifies w.r.t tvk. Note that $\mathcal{F}_1$ knows tsk and hence it can produce such a signature. By the witness-indistinguishability of NIZK_1 (implied by the zero-knowledge property) $\mathcal{B}$ cannot tell how the modified predicate was satisfied and hence cannot distinguish this signature from a real signature where the actual attributes of the user are used. The rest of $\mathcal{B}$'s queries are answered normally as in Fig. 4-1.

Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{F}_1$ uses the NIZK_1's extraction key xk_1 to extract the witness and returns the tagged signature on the identity and the attributes if the forgery involves the attribute authority i. Otherwise, it aborts. The probability that $\mathcal{F}_1$ guesses the correct authority is $\frac{1}{\kappa(\lambda)}$.

- **Adversary $\mathcal{F}_2$:** Adversary $\mathcal{F}_2$ gets tvk from its game and has access to an oracle Sign that it uses to obtain signatures that verify w.r.t. tvk on messages of its choice. It runs $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$ and $(\mathsf{crs}_2, \mathsf{xk}_2) \leftarrow \mathsf{NIZK}_2.\mathsf{Setup}(1^\lambda)$. It also creates the key pair $(\mathsf{esk}, \mathsf{epk})$ for the encryption scheme. It then forwards $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathsf{tvk}, \mathbb{A}, \mathcal{H})$ and $\mathsf{tk} = \mathsf{esk}$ to $\mathcal{B}$.

  When asked for AddA queries, $\mathcal{F}_2$ creates the authority keys itself. Whenever $\mathcal{B}$ asks AddS queries, $\mathcal{F}_2$ uses the corresponding authorities' secret keys $\mathsf{aask}_{\mathsf{aid}(a)}$ to create the key for the signer. When asked for Sign queries on $(\mathsf{id}, \mathcal{A}, m, \Psi)$, $\mathcal{F}_2$ queries its Sign oracle to obtain a signature on the corresponding pseudo-attribute matching (Ψ, m) and constructs the rest of the signature by encrypting id and generating the proof π using id and the signature on the pseudo-attribute

$a_{\Psi,m}$ as a witness. By the zero-knowledge of the proof system, the adversary cannot tell which witness was used in the proof. The rest of $\mathcal{B}$'s queries are answered normally as in Fig. 4-1.

Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{F}_2$ uses NIZK_1's extraction key xk_1 to extract the witness and returns the signature on the pseudo-attribute $a_{\hat{\Psi}^*,m^*}$ if $\mathcal{B}$ created the untraceable signature by using a forged signature on a new pseud-attribute that it did not get from querying $\mathcal{F}_2$; otherwise, $\mathcal{F}_2$ aborts.

□

4.9 Conclusion

In this chapter, we have presented our *Decentralized Traceable Attribute Based Signature scheme* (DTABS). We first gave the intuition of the scheme, defined the *decentralisation* and *traceability* notions. We then gave the syntax and security definitions of our DTABS. Moreover, we gave the modules needed to generically construct it. We then gave a couple of generic constructions, one instantiation in the symmetric groups, and two in the asymmetric ones. We ended the chapter by proving the theorems of DTABS.

Chapter 5

Attribute Based Signatures with User-Controlled Linkability

5.1 Introduction

In this chapter, we introduce Attribute-Based Signatures with User-Controlled Linkability (ABS-UCL) [EKCGD14]. This is a joint work with Liqun Chen (HP Labs Bristol), Essam Ghadafi (University of Bristol) and James Davenport (University of Bath).

Attribute-based signatures allow a signer who has enough credentials/attributes to anonymously sign a message with respect to some public policy revealing neither the attributes used nor his identity. *User-Controlled Linkability* (UCL) is a new feature which allows a user to make some of his signatures directed at the same recipient linkable while still retaining anonymity. Such a feature is useful for many real-life applications (see below). We give a general framework of an ABS-UCL and its security model, a standard model consideration and a *practical* construction that supports multiple attribute authorities.

UCL is a lightweight solution that allows a user to opt to make some of his signatures directed at the same verifier linkable without sacrificing anonymity. Unlike the reliance on tracing authorities, which are generally thought of as "for trouble-shooting", UCL is intended to be built into normal use. For example, in the world of attributes, assume that a signer wants to establish a session (in a analogous way to the idea of cookies in Web browsers) with a recipient and maintain this session in a con-

vincing way that he is indeed the same person whom the recipient is communicating with, not someone else who also has enough credentials to satisfy the same policy in question. A tracing authority cannot help here, whereas user-controlled linkability is an ideal functionality for such a scenario.

ABS-UCL **main contribution**

- A formal security model for attribute based signatures with user controlled linkability.
- Decentralization: Again, and similar to our DTABS, we deal with the case where there are multiple attribute authorities that are responsible for distributing the attributes/credentials to the users.
- User-Controlled Linkability: This allows for a signer to convince a certain verifier that a set of signatures belongs to him without revealing any further information about his identity or the attributes that he holds.
- Generic construction for ABS-UCL: We abstractly use the underlying modules needed to realise ABS-UCL. Security proofs are also given for the generic construction.
- Instantiations: We give a practical instantiation of ABS-UCL in the random oracle model. Moreover, we use the given instantiation to give a practical instantiation of our DTABS.

5.2 Syntax and Security Definitions of ABS-UCL

In this section, we define the notion of Attribute-Based Signatures with User-Controlled Linkability (ABS-UCL), and present its security requirements. Our notion supports multiple attribute authorities, each responsible for a subset of attributes.

5.2.1 Syntax of ABS-UCL

In an ABS-UCL scheme, we have a set $\mathbb{AA} = \{\mathsf{AA}_i\}_{i=1}^{n}$ of attribute authorities, where $\mathbb{A}_i$ is the space of attributes managed by attribute authority AA_i. The universe of attributes is defined as $\mathbb{A} = \bigcup_{i=1}^{n} \mathbb{A}_i$. Assume that $\mathcal{A} \subset \mathbb{A}$ is a set of attributes for which

a certain predicate Ω is satisfied, i.e. $\Omega(\mathcal{A}) = 1$. We have, $a \in \mathcal{A} \Rightarrow \exists \mathbb{A}_i$, s.t. $a \in \mathbb{A}_i$, so attribute a is managed by attribute authority AA_i. Below are the definitions of the algorithms used in an ABS-UCL scheme, where all algorithms (bar the first three) take as implicit input pp produced by Setup.

- $\mathsf{Setup}(1^\lambda)$: On input a security parameter, it returns public parameters pp.
- $\mathsf{AASetup}(\mathsf{aid}, \mathsf{pp})$: Is run locally by attribute authority $\mathsf{AA}_{\mathsf{aid}}$ to generate its public/secret key pair $(\mathsf{aavk}, \mathsf{aask})$. The authority publishes aavk and keeps aask secret.
- $\mathsf{UKeyGen}(\mathsf{id}, \mathsf{pp})$: Is run by user id to generate his personal secret key $\mathsf{sk}_{\mathsf{id}}$.
- $\mathsf{AttKeyGen}(\mathsf{id}, f(\mathsf{sk}_{\mathsf{id}}), a, \mathsf{aask})$: Is run by attribute authority AA that is responsible for the attribute a, where f is an injective one-way function, it gives the user id the secret key $\mathsf{sk}_{\mathsf{id},a}$, bound to his identity id and $f(\mathsf{sk}_{\mathsf{id}})$.
- $\mathsf{Sign}(m, \Omega, \mathsf{sk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id},\mathcal{A}}, \mathsf{recip})$: If a user has enough attributes to satisfy the predicate Ω, i.e. $\Omega(\mathcal{A}) = 1$, then he uses the corresponding secrets keys $\mathsf{sk}_{\mathsf{id},\mathcal{A}} = \{\mathsf{sk}_{\mathsf{id},a_i}\}_{a_i \in \mathcal{A}}$ to produce a valid signature $\sigma = \{\sigma_{\mathsf{ABS}}, \sigma_{\mathsf{UCL}}\}$ on the message m and the recipient tag recip w.r.t. the predicate Ω; if $\mathsf{recip} = \perp$ then $\sigma_{\mathsf{UCL}} = \perp$.
- $\mathsf{Verify}(\sigma, \{\mathsf{vk}_{\mathsf{AA}_i}\}_i, \Omega, m, \mathsf{recip})$: Takes a signature σ on the message m and the possibly empty recipient tag recip w.r.t. a predicate Ω, the verification keys $\{\mathsf{vk}_{\mathsf{AA}_i}\}_i$ of the attribute authorities managing attributes involved in Ω, and returns 1 if the signature is valid, and 0 otherwise.
- $\mathsf{Link}(\sigma_0, m_0, \{\mathsf{vk}_{\mathsf{AA}_i}\}_i, \Omega_0, \sigma_1, m_1, \{\mathsf{vk}_{\mathsf{AA}_j}\}_j, \Omega_1, \mathsf{recip})$: On input two signatures, two messages, two signing policies and the verification keys of the attribute authorities managing the attributes involved in the policies, and a recipient tag, it returns 1 if the signatures are valid on their respective messages and the same non-empty recipient tag recip (w.r.t. the respective policy), i.e. if $\mathsf{recip} \neq \perp$ and $(\sigma_{\mathsf{UCL}0} = \sigma_{\mathsf{UCL}1} \neq \perp)$, and 0 otherwise.

5.2.2 Security Definitions

We define here the security requirements of an ABS-UCL scheme.

Correctness. This requires that signatures produced by honest users verify correctly and that signatures produced by the same user to the same valid recipient (i.e. on the same non-empty recipient tag) link.

Linkability. As specified in [ISO13], there are two approaches to support user-controlled linkability in anonymous digital signatures: in the first, a designated linking authority can determine whether or not two signatures are linked; whereas in the second method, there exists a public linking algorithm which can be run by any party. Our model supports the latter. We require that only valid signatures directed at the same recipient and which were produced by the same user link. In the linkability game the adversary can choose all the secret keys of the users and attribute authorities. The adversary outputs $(\sigma_1, \mathsf{recip}_1, m_1, \{\mathsf{vk}_{\mathsf{AA}_i}\}_i, \Omega_1, \mathsf{sk}_1)$ and $(\sigma_2, \mathsf{recip}_2, m_2, \{\mathsf{vk}_{\mathsf{AA}_j}\}_j, \Omega_2, \mathsf{sk}_2)$. It wins if σ_i is valid (w.r.t. Ω_i) on m_i and recip_i, for $i = 1, 2$ and either of the following holds:

- σ_1 was produced by sk_1 and σ_2 was produced by sk_2 where $\mathsf{sk}_1 = \mathsf{sk}_2$ and $\mathsf{recip} = \mathsf{recip}_1 = \mathsf{recip}_2 \neq \perp$ but $\mathsf{Link}(\sigma_1, m_1, \{\mathsf{vk}_{\mathsf{AA}_i}\}_i, \Omega_1, \sigma_2, m_2, \{\mathsf{vk}_{\mathsf{AA}_j}\}_j, \Omega_2, \mathsf{recip}) = 0$.

- σ_1 was produced by sk_1 and σ_2 was produced by sk_2 where $\mathsf{sk}_1 = \mathsf{sk}_2$ and $\mathsf{Link}(\sigma_1, m_1, \{\mathsf{vk}_{\mathsf{AA}_i}\}_i, \Omega_1, \sigma_2, m_2, \{\mathsf{vk}_{\mathsf{AA}_j}\}_j, \Omega_2, \mathsf{recip}_k) = 1$ for $k \in \{1, 2\}$ and either $\mathsf{recip}_k = \perp$ or $\mathsf{recip}_1 \neq \mathsf{recip}_2$.

- σ_1 was produced by sk_1 and σ_2 was produced by sk_2 where $\mathsf{sk}_1 \neq \mathsf{sk}_2$ and $\mathsf{recip} = \mathsf{recip}_1 = \mathsf{recip}_2 \neq \perp$ and $\mathsf{Link}(\sigma_1, m_1, \{\mathsf{vk}_{\mathsf{AA}_i}\}_i, \Omega_1, \sigma_2, m_2, \{\mathsf{vk}_{\mathsf{AA}_j}\}_j, \Omega_2, \mathsf{recip}) = 1$.

In summary, this requires that signatures by the same user on the same non-empty recipient tag link. Also, signatures by different users but on the same recipient tag or those by the same user but on different recipient tags do not link.

Anonymity. This requires that a signature reveals neither the identity of the signer nor the attributes used in the signing. In the anonymity game, we have the following:

- Adversary's Capabilities: Full control over *all* attribute authorities. It can also ask for the secret keys of signers of its choice; those signers will be referred to as *corrupt* users. In addition, the adversary can ask for the secret key of any attribute and has a signing oracle that it can query on messages and recipient tags on behalf of honest users.

- Adversary's Challenge: The adversary outputs $(m, \mathsf{id}_0, \mathcal{A}_0, \mathsf{id}_1, \mathcal{A}_1, \Omega, \mathsf{recip})$ where $\Omega(\mathcal{A}_i) = 1$ for $i = 0, 1$. If $\mathsf{recip} \neq \perp$ then we require that throughout the game (i.e. even after the challenge phase) id_0 and id_1 must be honest (i.e. their personal secret keys are not revealed to the adversary), and that neither of $(\mathsf{id}_0, \mathsf{recip})$, $(\mathsf{id}_1, \mathsf{recip})$ is queried to the signing oracle. This ensures that the adversary cannot trivially win by exploiting the linkability feature.

  The adversary gets back a signature σ_b produced using $(\mathsf{id}_b, \mathcal{A}_b)$ for $b \leftarrow \{0, 1\}$. After this, the adversary can continue accessing its oracles as long as it does not violate the above two conditions.

- Adversary's Output: The adversary outputs its guess b^* and wins if $b^* = b$.

Unforgeability. This requires that users cannot output signatures on (message, recipient tag) pairs w.r.t. to a signing policy not satisfied by their set of attributes, even if they pool their attributes together, which ensures collusion-resistance. In addition, since our notion supports user-controlled linkability, we additionally require that an adversary cannot produce signatures which link to other signatures by an honest user, i.e. one whose personal secret key has not been revealed to the adversary, even if all other users and attribute authorities in the system are corrupt. Note that, unlike in DAA, e.g. [BFG13a, BFG$^+$13b], in our notion even if a user's personal secret key is revealed, only signatures on non-empty recipient tags by the user can be traced, i.e. it is impossible to trace signatures on empty recipient tags.

In the unforgeability game, we have the following:

- Adversary's Capabilities: Access to a signing oracle. Moreover, it can corrupt any attribute authority. We refer to the non-corrupted attribute authorities as *honest* ones. It can also ask for the personal secret key of any user. We refer to the non-corrupted users also as honest ones. It can also ask for the secret key for any attribute.

- Winning Conditions: The adversary wins if either:

  – Adversary outputs a valid signature σ on m and recip w.r.t. Ω, where $(m, \mathsf{recip}, \Omega)$ was not queried to the signing oracle, and there exists no subset of attributes $\mathcal{A}^*$ whose keys have been revealed to the adversary or managed by corrupt attribute authorities s.t. $\Omega(\mathcal{A}^*) = 1$. In other words,

$\forall \mathcal{A}^*$ s.t. $\Omega(\mathcal{A}^*) = 1, \exists a^* \in \mathcal{A}^*$ s.t. $\Omega(\mathcal{A}^* \setminus \{a^*\}) = 0$ and a^*'s key has never been revealed to the adversary and it is managed by an honest attribute authority.

– Adversary outputs a tuple $(m_0, \sigma_0, \{\mathsf{vk}_{\mathsf{AA}_i}\}_i, \Omega_0, m_1, \sigma_1, \{\mathsf{vk}_{\mathsf{AA}_j}\}_j, \Omega_1,$ $\mathsf{recip} \neq \perp, \mathsf{id})$, where σ_0 is valid on m_0 and recip w.r.t. Ω_0, σ_1 is valid on m_1 and recip w.r.t. Ω_1, user id is honest, $\mathsf{Link}(\sigma_0, m_0, \{\mathsf{vk}_{\mathsf{AA}_i}\}_i, \Omega_0, \sigma_1, m_1,$ $\{\mathsf{vk}_{\mathsf{AA}_j}\}_j, \Omega_1, \mathsf{recip}) = 1$ and either $(\mathsf{id}, m_0, \mathsf{recip}, \Omega_0)$ or $(\mathsf{id}, m_1, \mathsf{recip}, \Omega_1)$ was not queried to the signing oracle.

Note here the adversary has more freedom than it has in the anonymity game because it is allowed to ask for signatures by the honest user it intends to frame on any recipient tag.

5.3 Modules needed to Construct ABS-UCL

5.3.1 Bilinear Groups

We refer the reader to Sec. 3.5 for information on bilinear maps. In our construction, we use type-3 pairings.

5.3.2 Digital Signatures

A *digital signature* for a message space $\mathcal{M}$ is a tuple of polynomial-time algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, where KeyGen outputs a pair of secret/verification keys $(\mathsf{sk}, \mathsf{vk})$ for the signer; $\mathsf{Sign}(\mathsf{sk}, m)$ outputs a signature σ on the message m; $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$ outputs 1 if σ is a valid signature on the message m or 0 otherwise.

Besides correctness, the security of a digital signature requires existential unforgeability under an adaptive chosen-message attack which demands that all PPT adversaries $\mathcal{F}$ have a negligible advantage in winning the following game:

- A key pair $(\mathsf{sk}, \mathsf{vk})$ is generated and vk is sent to $\mathcal{F}$.
- Adversary $\mathcal{F}$ makes a polynomial number of queries to a sign oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$.
- Eventually, $\mathcal{F}$ halts by outputting (σ^*, m^*) and wins if σ^* is valid on m^*, and m^* was not queried to Sign.

A weaker variant of existential unforgeability (i.e. existential unforgeability under a *weak* chosen-message attack) requires that the adversary sends all its queries before seeing the verification key.

We use different variants of the full Boneh-Boyen signature scheme [BB04a]. We refer to original full Boneh-Boyen scheme as the BB scheme, whereas we refer to its modified variant originally defined in [BB04a], and used in, e.g. [Che10], as the BB† scheme. Both schemes are secure under the q-SDH assumption.

Let $\mathcal{P} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ be the description of a bilinear group and $h_1 \in \mathbb{G}_1$ is a random element. The schemes are described below:

- $\mathsf{KeyGen}(\mathcal{P})$: Choose $x, y \leftarrow \mathbb{Z}_p$, set $(X, Y) = (g_2^x, g_2^y)$. The secret key is (x, y) and the verification key is (X, Y).

- $\mathsf{BB.Sign}(\mathsf{sk}, m)$: To sign $m \in \mathbb{Z}_p$, choose $r \leftarrow \mathbb{Z}_p$ such that $x + ry + m \neq 0$ and compute the signature $\sigma = g_1^{1/(x+ry+m)}$. In the BB† scheme, the signature is $\sigma = (g_1 \cdot h_1^z)^{1/(x+ry+m)}$, where the BB† signer need not know the value z.

- $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$: if $e(\sigma, X \cdot Y^r \cdot g_2^m) = e(g_1, g_2)$ output 1, otherwise 0.
  In the BB† scheme, the verification equation is $e(\sigma, X \cdot Y^r \cdot g_2^m) = e(g_1 \cdot h_1^z, g_2)$

5.3.3 Linkable Indistinguishable Tag Scheme LIT

A *Linkable Indistinguishable Tag* (LIT) scheme [BFG+13b] is similar to a Message Authentication Code (MAC) but requires different security properties. It consists of a couple of algorithms KeyGen and Tag. The former, on input a security parameter, produces a secret key sk, whereas the latter, on input a message m and the secret key, outputs a tag.

Besides correctness, the security of LIT [BFG+13b] requires Linkability and f-Indistinguishability. Linkability requires that an adversary who is allowed to control both the secret key and the message cannot produce equal tags unless they are tags on the same message/key pair. Indistinguishability, which is defined w.r.t. a one-way function f of the secret key, requires that an adversary who gets $f(\mathsf{sk})$ and access to a tag oracle, cannot determine whether or not a new tag on a message of its choice was produced using the same key used by the tag oracle.

As in [BFG+13b], we instantiate the LIT in the ROM with the Boneh-Lynn-Shacham (BLS) signature scheme [BLS04b]. The LIT instantiation is secure under

the DDH and the discrete logarithm problem Dlog [BFG+13b].

5.3.4 Non-Interactive Zero-Knowledge Proofs

We refer the reader to to the formal definitions of NIZK systems that are given in Sec 3.8.2. In our construction in the random oracle model, we use the Fiat–Shamir transformation [FS87] applied to interactive Σ-protocols.

5.3.5 Span Programs

We refer the reader to Sec 4.3.7, in which we define Span programs and give an example to show how it works.

5.4 Generic Construction of ABS-UCL

Overview of the Framework.The tools we use in our generic construction are: a NIZK system NIZK that is sound and zero-knowledge, two existentially unforgeable signature schemes DS_1 and DS_2, a collision-resistant hash function $\mathcal{H}$ and a $f-$ indistinguishable linkable indistinguishable tag scheme LIT. The Setup algorithm of ABS-UCL generates the common reference string crs for the NIZK system NIZK. It also generates a key pair $(\mathsf{vk}_{\mathsf{psdo}}, \mathsf{sk}_{\mathsf{psdo}})$ for the digital signature schemes DS_2. The public parameters of the system are set to $\mathsf{pp} = (\mathsf{crs}, \mathsf{vk}_{\mathsf{psdo}}, \mathbb{A}, \mathcal{H})$, where $\mathbb{A}$ is the universe of attributes. For a new attribute authority to join the system, it creates a secret/verification key pair $(\mathsf{sk}_{\mathsf{aid}}, \mathsf{vk}_{\mathsf{aid}})$ for signature scheme DS_1. To generate a signing key for attribute $a \in \mathbb{A}$ for signer id, the managing attribute authority signs the signer identity along with the attribute and the image of the one-way function on his secret key, i.e. $(\mathsf{id}, a, f(\mathsf{sk}_{\mathsf{id}}))$, using $\mathsf{sk}_{\mathsf{aid}}$. The resulting signature is used as the secret key for that attribute by signer id.

To sign a message m w.r.t. a signing policy Ω, there are two cases;

- If the signature is linkable (i.e. on a non-empty recipient tag $\mathsf{recip} \neq \perp$), the signer first uses LIT and his secret key to compute a tag σ_{UCL} on the recipient name recip and a NIZK proof π that such a tag verifies w.r.t. his personal secret key $\mathsf{sk}_{\mathsf{id}}$, and that he either has a digital signature on a pseudo-attribute (following [MPR11, EKGK13]), i.e. the hash of the combination of the signing

predicate, the message and the recipient name recip, i.e. $a_{\mathsf{psdo}} = \mathcal{H}(\Omega, m, \mathsf{recip})$, that verifies w.r.t. the verification key $\mathsf{vk}_{\mathsf{psdo}}$ or that she has enough credentials (DS_1 signatures on $(\mathsf{id}, f(\mathsf{sk}_{\mathsf{id}}), a_i)$) to satisfy the original signing predicate Ω.

- For non-linkable signatures (i.e. when recip $=\perp$), it suffices to produce a NIZK proof that the signer has enough attributes to satisfy the modified predicate, i.e $\hat{\Omega} = \Omega \vee a_{\mathsf{psdo}}$, and therefore, no need for the linking part that uses LIT. Note that in this case $a_{\mathsf{psdo}} = \mathcal{H}(\Omega, m)$.

Before we define the languages for the NIZK proofs $\mathcal{L}_1$ for linkable and $\mathcal{L}_2$ for non-linkable signatures, we will generically define the format of these languages, where the secret values, aka witnesses for proofs, are underlined:

$$\mathcal{L} : \Big\{(\text{public values} \quad \mathsf{pv}), (\text{witness} \quad \underline{\mathsf{w}}) : \mathsf{R_i}(\mathsf{pv}, \underline{\mathsf{w}})\Big\}$$

- **Linkable signatures** ($\mathsf{recip} \neq \perp$)**:**

$$\mathcal{L}_1 : \Big\{\big((\vec{\mathsf{vk}} = \{\mathsf{vk}_i\}_{i=1}^{|\hat{\Omega}|}, \vec{a} = \{a_i\}_{i=1}^{|\hat{\Omega}|}), (\underline{\mathsf{sk}_{\mathsf{id}}}, \underline{\mathsf{id}}, \underline{\vec{v}}, \underline{\vec{\sigma}} = \{\sigma_{a_i}\}_{i=1}^{|\hat{\Omega}|})\big) :$$
$$\Big(\underline{\vec{v}}\mathbf{Z} = [1, 0, \ldots, 0]\Big) \bigwedge_{i=1}^{|\hat{\Omega}|-1} \Big(\underline{v_i} = 0 \vee \mathsf{DS}_1.\mathsf{Verify}(\mathsf{vk}_i, \underline{\mathsf{id}}, \underline{\mathsf{sk}_{\mathsf{id}}}, a_i, \underline{\sigma_{a_i}}) = 1\Big)$$
$$\bigwedge \Big(\underline{v_{|\hat{\Psi}|}} = 0 \vee \mathsf{DS}_2.\mathsf{Verify}(\mathsf{vk}_{\mathsf{psdo}}, a_{\mathsf{psdo}}, \underline{\sigma_{a_{\mathsf{psdo}}}}) = 1\Big)$$
$$\bigwedge \Big(\mathsf{LIT}.\mathsf{Tag}(\underline{\mathsf{sk}_{\mathsf{id}}}, \mathsf{recip}) = \sigma_{\mathsf{UCL}}\Big\}.$$

- **Non-Linkable signatures** ($\mathsf{recip} = \perp$)**:**

$$\mathcal{L}_2 : \Big\{\big((\vec{\mathsf{vk}} = \{\mathsf{vk}_i\}_{i=1}^{|\hat{\Omega}|}, \vec{a} = \{a_i\}_{i=1}^{|\hat{\Omega}|}), (\underline{\mathsf{sk}_{\mathsf{id}}}, \underline{\mathsf{id}}, \underline{\vec{v}}, \underline{\vec{\sigma}} = \{\sigma_{a_i}\}_{i=1}^{|\hat{\Omega}|})\big) :$$
$$\Big(\underline{\vec{v}}\mathbf{Z} = [1, 0, \ldots, 0]\Big) \bigwedge_{i=1}^{|\hat{\Omega}|-1} \Big(\underline{v_i} = 0 \vee \mathsf{DS}_1.\mathsf{Verify}(\mathsf{vk}_i, \underline{\mathsf{id}}, \underline{\mathsf{sk}_{\mathsf{id}}}, a_i, \underline{\sigma_{a_i}}) = 1\Big)$$
$$\bigwedge \Big(\underline{v_{|\hat{\Psi}|}} = 0 \vee \mathsf{DS}_2.\mathsf{Verify}(\mathsf{vk}_{\mathsf{psdo}}, a_{\mathsf{psdo}}, \underline{\sigma_{a_{\mathsf{psdo}}}}) = 1\Big)$$

We use a span program to prove the satisfiability of the extended predicate $\hat{\Omega}$. Using a public matrix $\mathbf{Z}$, the signer needs to prove the ownership of a *secret* vector $\vec{v} \in \mathbb{Z}_p^{|\hat{\Omega}|}$ for which $\vec{v}\mathbf{Z} = [1, 0, \ldots, 0]$. The zero elements in this vector $\vec{v}$ corresponds to attributes that the signer does not actually need in order to satisfy the

predicate. For these values, the signer can safely choose random signatures. For the non-zero elements in $\vec{v}$, the signer needs to prove ownership of their corresponding attributes/pseudo-attribute.

The hiding property of the NIZK system ensures that the proof π does not reveal how the modified predicate $\hat{\Omega}$ was satisfied.

The pseudo-attribute is used for two reasons; firstly, it binds the signature to the message, the signing predicate, and the recipient name recip if the the signature is linkable. Secondly, the secret signing key $\mathsf{sk}_{\mathsf{psdo}}$ for the digital signature scheme DS will be used as a trapdoor in the security proofs to allow its holder to simulate signatures and sign on behalf of any signer without knowing their secret keys. That could be done by producing a signature on the pseudo-attribute associated with the message and the signing predicate.

The full proof for the following Theorem is in 5.6.

Theorem 7. *The generic construction of the attribute-based signature with user- controlled linkability* ABS-UCL *given above is secure if the underlying building blocks are secure.*

5.5 A Practical Instantiation of ABS-UCL

Description of the Construction. The signer's task is to provide a zero-knowledge proof of knowledge π w.r.t. the languages defined earlier, i.e. $\mathcal{L}_1$ and $\mathcal{L}_2$, depending on whether or not the signature is linkable. We instantiate DS_1 using the BB† scheme and DS_2 using the BB scheme. The proof will be made of 3 parts (or 2 if non-linkable). The first deals with the Span program to show how to hide which subset of attributes the signer has used to satisfy the modified predicate $\hat{\Omega}$. For this, the signer proves that he has used a secret vector $\vec{v}$ to span the public matrix $\mathbf{Z} \in \mathbb{Z}_p^{\alpha \times \theta}$ of the span program, where $\alpha = |\hat{\Omega}|$. The second part is to show that the signatures verify correctly w.r.t. their corresponding verification keys, where the span program can safely let the signer choose random signatures for the attributes which he does not own/want to use. The third part is to show that, when the signature is supposed to be linkable, the linking part indeed uses the same user secret key used in the rest of the proof. Not that the group elements used later in the commitments, i.e. k_1, k_2 and k_3 are parts of the public parameters pp whereas sk is the signer's secret key.

Part 1: Span program Prove that $\underline{\vec{v}}\mathbf{Z} = [1, 0, \ldots, 0]$. This can be done by proving the following:

$$\sum_{i=1}^{\alpha} v_i \mathbf{Z}_{ij} = \begin{cases} 1 & j = 1 \\ 0 & 2 \leq j \leq \theta \end{cases} \tag{5.1}$$

- Commitments of vector $\vec{v}$
  - $\beta_{v_i}, \beta_{t_i}, t_i \leftarrow \mathbb{Z}_p, i = 1 \ldots \alpha.$
  - $\mathcal{V}_i = g_1^{\beta_{v_i}} \cdot k_3^{\beta_{t_i}}; \quad \hat{v_i} = g_1^{v_i} \cdot k_3^{t_i}$
- Proof of Statement
  - $\forall j \in [1, \theta]$ compute: $\Lambda_j = \prod_{i=1}^{\alpha} k_3^{t_i . M_{ij}}; \quad \lambda_j = \prod_{i=1}^{\alpha} (k_3^{M_{ij}})^{\beta_{t_i}}$

Part 2: DS_1 and DS_2 Now each verification equation is as follows:

$$e(\underline{\sigma_{a_i}}^{v_i}, X \cdot Y^{\underline{r}} \cdot g_2^{a_i || \underline{\mathsf{id}}}) = e(g_1, g_2) \cdot e(h_1^{\underline{\mathsf{sk}}}, g_2)$$

DS_1 is instantiated using the BB† scheme whereas DS_2 is instantiated using the BB scheme. The signatures are as follows:

$$\sigma_{a_i} = \begin{cases} (g_1 \cdot h_1^{\mathsf{sk}})^{1/(x_i + y_i r_i + a_i || \mathsf{id})} & \text{regular attributes} \\ g_1^{1/(x_i + y_i r_i + a_{\mathsf{psdo}})} & \text{pseudo-attributes} \end{cases}$$

Where the public keys of an attribute a_i is the couple of group elements $X_i = g_2^{x_i}$ and $Y_i = g_2^{y_i}$. The identity of the signer is id and his secret key is sk. In order to use the secret vector $\vec{v}$ to hide the subset of attributes used to satisfy the predicate Ω, we can simply raise each σ_{a_i} to its corresponding vector value v_i. When v_i is zero, the signer does not want to use this attribute, and therefore he can replace the signature by a random value.

- Commitments of $(\sigma_{a_i}, r_i), i \in [1, \alpha]$ and the signer identity id:

  Pick $\rho_{v_i}, \rho_{\mathsf{id}}, \rho_{r_i}, \rho_{\mathsf{sk}}, \beta_{\rho_{\mathsf{sk}}}, \beta_{\mathsf{id}\rho_{v_i}}, \beta_{r_i}, \beta_{\rho_i}, \beta_{\mathsf{id}}, \beta_{\rho_{r_i}}, \beta_{\rho_{\mathsf{id}}}, \beta_{cs}, \leftarrow \mathbb{Z}_p$, and compute:

$$T_i = {\sigma_{a_i}}^{v_i} \cdot {k_1}^{\rho_{v_i}}, \quad K_i = Y^{r_i} \cdot {k_2}^{\rho_{r_i}}, \quad Z = h_1^{\mathsf{sk}} \cdot k_1^{\rho_{\mathsf{sk}}} \quad U = {g_2}^{\mathsf{id}} \cdot {k_2}^{\rho_{\mathsf{id}}}$$

$$\hat{K}_i = {Y_i}^{\beta_{r_i}} \cdot {k_2}^{\beta_{\rho r_i}}, \quad \hat{Z} = h_1^{\beta_{\mathsf{sk}}} \cdot k_1^{\beta_{\rho_{\mathsf{sk}}}}, \quad \hat{U} = {g_2}^{\beta_{\mathsf{id}}} \cdot {k_2}^{\beta_{\rho_{\mathsf{id}}}}$$

Let, $\forall i \in [1, \alpha - 1] : \rho_i = \rho_{r_i} + \rho_{\mathsf{id}}$ whereas $\rho_\alpha = \rho_{r_\alpha}$.

- Simplification: (can be done by both prover and verifier)

$$X'_i = e(k_1, X_i \cdot g_2^{a_i \cdot 2^{|\mathsf{id}|}}) \quad Y'_i = e(k_1, Y_i) \quad R = e(k_1, g_2)$$
$$T'_i = e(T_i, k_2) \quad D' = e(k_1, g_2^{a_{\mathsf{psdo}}})$$

- Knowledge of Exponents
  $\forall i \in [1, \alpha]$ and $\forall j \in [1, \theta]$, compute:

$$\mathcal{X'}_{ij} = ({X'_i}^{M_{ij}})^{\beta_{\rho v_i}} \quad \mathcal{Y'}_{ij} = ({Y'_i}^{M_{ij}})^{\beta_{r_i \rho v_i}} \quad \mathcal{T'}_{ij} = ({T'_i}^{M_{ij}})^{\beta_{\rho_i}}$$

  $\forall i \in [1, \alpha - 1]$, $\forall j \in [1, \theta]$, compute:

$$\mathcal{R}_{ij} = (R^{M_{ij}})^{\beta_{\mathsf{id}} \rho_{v_i}}$$

  $\forall j \in [1, \theta]$:

  - $\mathcal{D'}_{\alpha j} = ({M'}^{z_{\alpha j}})^{\beta_{\rho v_i}}$
  - $\mathcal{P}_j = \mathcal{X'}_{\alpha j} \cdot \mathcal{Y'}_{\alpha j} \cdot \mathcal{T'}_{\alpha j} \cdot \mathcal{D'}_{\alpha j}$
  - $\mathcal{B}_j = \mathcal{P}_j \cdot \prod_{i=1}^{\alpha-1} \mathcal{X'}_{ij} \cdot \mathcal{Y'}_{ij} \cdot \mathcal{R}_{ij} \cdot \mathcal{T'}_{ij}$

Part 3:Linkability- LIT The signer needs to prove the following equation:

$$\mathsf{BLS.Sign}(\underline{\mathsf{sk}}, \mathsf{recip}) = \sigma_{\mathsf{UCL}}$$

If the signature is linkable, then compute:

$$\mathcal{N} = \mathcal{H}(\mathsf{recip})^{\beta_{\mathsf{sk}}}, \quad \mathcal{L} = \Big(\frac{h_1}{H(\mathsf{recip})}\Big)^{\beta_{\mathsf{sk}}} \cdot k_1^{\beta_{\rho_{\mathsf{sk}}}} \quad \sigma_{\mathsf{UCL}} = \mathcal{H}(\mathsf{recip})^{\mathsf{sk}},$$

otherwise; $\sigma_{\mathsf{UCL}} = \perp$.

Finally, compute the challenge c:

$$c = \mathcal{H}_{\mathsf{FS}}(\underbrace{\mathcal{N}||\mathcal{L}}_{\text{if linkable}} ||\lambda_j||\mathcal{S}_i||T_i||K_i||U||\hat{K}_i||\hat{U}||\mathcal{B}_j||Z), \forall i \in [1,\alpha], \forall j \in [1,\theta].$$

- Responses
  - $\mathfrak{r}_{v_i} = \beta_{v_i} + cv_i$, $\mathfrak{r}_{t_i} = \beta_{t_i} + ct_i$, $\mathfrak{r}_{\mathsf{id}} = \beta_{\mathsf{id}} + c\mathsf{id}$, $\quad s_{\mathsf{sk}} = \beta_{\mathsf{sk}} + c\mathsf{sk}$, $\quad s_{\rho_{\mathsf{sk}}} = \beta_{\rho_{\mathsf{sk}}} + c\rho_{\mathsf{sk}}$, $\quad \mathfrak{r}_{\rho_{\mathsf{id}}} = \beta_{\rho_{\mathsf{id}}} + c\rho_{\mathsf{id}}$
  - $\forall i \in [1,\alpha]$:
    $\mathfrak{r}_{\rho_{v_i}} = \beta_{\rho_{v_i}} + c\rho_{v_i}$, $\mathfrak{r}_{r_i\rho_{v_i}} = \beta_{r_i\rho_{v_i}} + c(r_i\rho_{v_i})$, $\mathfrak{r}_{\rho_i} = \beta_{\rho_i} + c\rho_i$, $\mathfrak{r}_{r_i} = \beta_{r_i} + cr_i$,
    $\mathfrak{r}_{\rho_{r_i}} = \beta_{\rho_{r_i}} + c\rho_{r_i}$;
  - $\forall i \in [1,\alpha-1]$, compute:
    $\mathfrak{r}_{\mathsf{id}\rho_{v_i}} = \beta_{\mathsf{id}\rho_{v_i}} + c(\mathsf{id}\rho_{v_i})$

Let $\Sigma = \{\mathfrak{r}_{\rho_{v_i}}, \mathfrak{r}_{r_i\rho_{v_i}}, \mathfrak{r}_{\mathsf{id}_i}, \mathfrak{r}_{\rho_i}, \mathfrak{r}_{r_i}, \mathfrak{r}_{\rho_{r_i}}, \mathfrak{r}_{\mathsf{id}}, \mathfrak{r}_{\rho_{\mathsf{id}}}, \mathfrak{r}_{v_i}, \mathfrak{r}_{t_i}, s_{\mathsf{sk}}, s_{\rho_{\mathsf{sk}}}\}$, the signature is:

$$\sigma_{\mathsf{ABS-UCL}} = (\Sigma, c, \{\Lambda_j\}_1^\theta, \{\hat{v}_i, T_i, K_i\}_1^\alpha, U, Z, \sigma_{\mathsf{UCL}})$$

Verification

Compute:

$$\Delta_j = e(T_\alpha, (X_\alpha \cdot K_{1\alpha} \cdot g_2^{a_{\mathsf{psdo}}})^{M_{\alpha j}})$$

$$E_j = \begin{cases} \Delta_1 \cdot \prod_{i=1}^{\alpha-1} e(T_i, (X_i \cdot K_i \cdot U)^{M_{ij}}) / e(g_1, g_2) \cdot e(Z, g_2) & j = 1 \\ \Delta_j \cdot \prod_{i=1}^{\alpha-1} e(T_i, (X_i \cdot K_i \cdot U)^{M_{ij}}) & 2 \le j \le \theta \end{cases}$$

- $\hat{U} = g_2^{\mathfrak{r}_{\mathsf{id}}} \cdot k_2^{\mathfrak{r}_{\rho_{\mathsf{id}}}} \cdot U^{-c}, \quad \hat{Z} = h_1^{s_{\mathsf{sk}}} \cdot k_1^{s_{\rho_{\mathsf{sk}}}} \cdot Z^{-c}$
- $\forall i \in [1,\alpha]$:
  $\mathcal{S}_i = g_1^{\mathfrak{r}_{v_i}} \cdot k_3^{\mathfrak{r}_{t_i}} \cdot \hat{v}_i{}^{-c} \quad \hat{K}_i = Y_i^{\mathfrak{r}_{r_i}} \cdot k_2^{\mathfrak{r}_{\rho_{r_i}}} \cdot K_i{}^{-c}$
- $\forall j \in [1,\theta]$:
  - $\lambda_j = \Lambda_j^{-c} \cdot \prod_{i=1}^{\alpha} (k_3^{M_{ij}})^{\mathfrak{r}_{t_i}}$

- $\mathcal{P}_j = (X'^{M_{\alpha j}}_{\alpha})^{\mathfrak{r}_{\kappa_\alpha}} \cdot (Y'^{M_{\alpha j}}_{\alpha})^{\mathfrak{r}_{r_\alpha \kappa_\alpha}} \cdot (T'^{M_{\alpha j}}_{\alpha})^{\mathfrak{r}_{\rho_\alpha}} \cdot (D'^{M_{\alpha j}})^{\mathfrak{r}_{\kappa_\alpha}}$
- $\mathcal{B}_j = E_j^{-c} \cdot \mathcal{P}_j \cdot \prod_{i=1}^{\alpha-1} (X'^{M_{ij}}_{i})^{\mathfrak{r}_{\rho_{v_i}}} \cdot (Y'^{M_{ij}}_{i})^{\mathfrak{r}_{r_i \rho_{v_i}}} \cdot (R^{M_{ij}})^{\mathfrak{r}_{\mathsf{id}_i}} \cdot (T'^{M_{ij}}_{i})^{\mathfrak{r}_{\rho_i}}$

- For the linkablility:
  - If $\sigma_{\mathsf{UCL}} \neq \perp$, then compute:

$$\mathcal{N} = \mathcal{H}(\mathsf{recip})^{\mathfrak{r}_{\mathsf{sk}}} \cdot (\sigma_{\mathsf{UCL}})^{-c}, \quad \mathcal{L} = \Big(\frac{h_1}{H(\mathsf{recip})}\Big)^{s_{\mathsf{sk}}} \cdot k_1^{s_{\rho_{\mathsf{sk}}}} \cdot \Big(\frac{Z}{\sigma_{\mathsf{UCL}}}\Big)^{-c}$$

- Let $\hat{c} = \mathcal{H}_{\mathsf{FS}}(\underbrace{\mathcal{N}||\mathcal{L}}_{\text{if linkable}} ||\lambda_j||\mathcal{S}_i||T_i||K_i||U||\hat{K}_i||\hat{U}||\mathcal{B}_j||Z)$,
- Verify that $\hat{c} = c$ and that the following statement holds:

$$\prod_{i=1}^{\alpha} \hat{v_i}^{M_{ij}} = \begin{cases} g_1 \cdot \Lambda_1 & j = 1 \\ \Lambda_j & 2 \leq j \leq \theta \end{cases}$$

The full proof for the following Theorem is in section 5.6 .

Theorem 8. *The construction is secure in the random oracle model if the q-SDH, DDH and Dlog assumptions hold, and the hash function $\mathcal{H}$ is collision resistant.*

5.6 Proofs

Correctness can be easily checked, it's implied by the building blocks' correctness.

Theorem 9. *Anonymity:*
If the NIZK proof system NIZK *is zero-knowledge, the linkable indistinguishable tag scheme* LIT *is indistinguishable, and the hash function $\mathcal{H}$ is collision-resistant then the generic construction is anonymous.*

Proof. We will show that if there exists an adversary $\mathcal{C}$ which breaks the anonymity of the ABS-UCL, we can construct adversaries $\mathcal{D}_1$ against the ZK property of the NIZK proof system NIZK, $\mathcal{D}_2$ against the indistinguishability of LIT.

$$\mathsf{Adv}^{\text{Anon}}_{\mathsf{ABS\text{-}UCL},\mathcal{C}}(\lambda) \leq \iota(\lambda) \cdot \left[2 \cdot \mathsf{Adv}^{\text{ZK}}_{\mathsf{NIZK},\mathcal{D}_1}(\lambda) + \mu(\lambda) \cdot \mathsf{Adv}^{\text{IND}}_{\mathsf{LIT},\mathcal{D}_2}(\lambda)\right]$$

Let $\iota(\lambda)$ and $\mu(\lambda)$, polynomials in λ, be the upper bounds of number of users that an adversary can create and the number of signing queries that he can ask for. We start the game by assuming the adversary $\mathcal{C}$ picks a specific user id in his challenger query, if he doesn't, the game aborts. Now, the adversary $\mathcal{D}_1$ gets the crs of the NIZK system NIZK from NIZK's challenger. He can honestly set up the rest of building blocks of the scheme himself. $\mathcal{D}_1$ needs to send a witness to the ZK challenger, he then gets back a proof π, his goal is to tell whether this proof is a real or a simulated one. The adversary $\mathcal{D}_1$ can answer $\mathcal{C}$'s signing queries by first producing a valid DS_1 signature for the user in question, and then use a random key sk to simulate LIT. He uses same random values for already queried couples (signer, recipient) since LIT is deterministic. When $\mathcal{D}_1$ gets his challenge input from $\mathcal{C}$, it picks one of the (honest) ids i.e.(id_1, id_2) sent by $\mathcal{C}$, and sends it to the NIZK challenger and gets back the proof π. It constructs the rest of the signature, i.e. σ_{UCL} using a random key to sign the recip. In the second game, and in order to show how reduce the forgery to the indistinguishability of the LIT, we will have a series of sub-games, where each two consecutive sub-games are indistinguishable by the indistinguishability of the LIT, i.e. they differ from each other by a single construction of a LIT tag. We start by answering all queries related to the selected signer id using its secret key (i.e. the one used in $f(\mathsf{sk}_{\mathsf{id}})$), then we move from a sub-game to another by answering one of these queries using a random key sk. We end up in the last sub-game where we answer all those queries using a key sk chosen uniformly at random. We have $\mu(\lambda)$ sub-games, where each two consecutive sub-games differ from each other by a negligible value, i.e. the advantage against the indistinguishability of the LIT. One can easily see that the last game is independent of the challenge bit b used in answering $\mathcal{C}$'s challenge query, and hence the anonymity of ABS-UCL. □

Theorem 10. *Unforgeability:*
The generic construction is unforgeable if the NIZK proof system NIZK *is sound, the hash function* $\mathcal{H}$ *(used in encoding pseudo-attributes) is collision-resistant, the* LIT *is linkable, and the underlying digital signature schemes* DS_1 *and* DS_2 *are existentially unforgeable*

Proof. We show that if an adversary $\mathcal{C}$ against the anonymity of the ABS-UCL exists, then we can construct a set of different adversaries, $\mathcal{D}_1$ against the soundness of the NIZK NIZK, $\mathcal{D}_2$ against the linkability property of the LIT, $\mathcal{D}_3$ against the unforge-

ability of the DS_1, $\mathcal{D}_4$ against the unforgeability of DS_2 and $\mathcal{D}_5$ against the collusion-resistance of the hash function $\mathcal{H}$ for which we have:

$$\begin{aligned}\mathsf{Adv}^{\text{Unforg}}_{\mathsf{ABS-UCL},\mathcal{C}}(\lambda) \leq & \mathsf{Adv}^{\text{sound}}_{\mathsf{NIZK},\mathcal{D}_1}(\lambda) + \nu(\lambda) \cdot \mathsf{Adv}^{\text{Unforg}}_{\mathsf{DS}_1,\mathcal{D}_3}(\lambda) + \mathsf{Adv}^{\text{Unforg}}_{\mathsf{DS}_2,\mathcal{D}_4}(\lambda) \\ & + \mathsf{Adv}^{\text{Coll}}_{\mathcal{H},\mathcal{D}_5}(\lambda)\end{aligned}$$

We let $\nu(\lambda)$, a polynomial in λ, be the upper bound of honest attribute authorities in this game. First, by the soundness of the NIZK NIZK, the adversary has negligible probability to successfully fake proofs for false statements. Moreover, any forging scenario that involves using different pairs of (message, predicate) that hash to same value would be directly used to break the collision-resistance property of the hash function $\mathcal{H}$. For the digital signatures DS_1 and DS_2, we deal with them in different cases. We will explain one of them and the second can be done analogously. For DS_1, the adversary $\mathcal{D}_3$ sets up the NIZK NIZK and DS_2, and then he makes a guess about an attribute authority AA_i, with probability of success $1/\nu(\lambda)$, that the adversary $\mathcal{C}$ will include an attribute managed by it. The adversary $\mathcal{D}_3$ has full control over the rest of the attribute authorities and thus he can deal with any request concerning these attribute authorities, but for attributes managed by the AA_i he forwards the queries to its challenger (i.e. DS_1's challenger). For Sign queries, he can use the pseudo-attribute to sign such queries. Eventually, when $\mathcal{D}_3$ receives the signature from $\mathcal{C}$, he can use the extraction key of the NIZK NIZK, which he has already set up, to extract the witnesses, and respond with the part concerning the guessed attribute authority AA_i to the DS_1's challenger. The game aborts if $\mathcal{D}_3$'s guess was not correct. The reduction to DS_2 can be done in a similar fashion, except that the adversary $\mathcal{D}_4$ doesn't need to make a guess about the attribute authority since the target is the pseudo-attribute, so in this case the game aborts if the forgery wasn't against it. □

Theorem 11. *User-controlled linkability:*
The attribute based signatures is User controlled linkable if the Linkable Indistinguishable Tag scheme LIT *is linkable.*

Proof. We will first deal with the case in which an adversary produces two supposedly linkable signatures, but when testing them with Link, it says they are not. Given that an adversary $\mathcal{C}$ has full control over the secret keys so he can generate secret keys to any identity that he wants to be challenged on, say $\mathsf{id}_{\mathsf{Link}}$. He should also pick the

verifier's name recip as a part of the challenge. At the end, he needs to produce two signatures, σ_1 and σ_2 on behalf of the user $\mathsf{id}_{\mathsf{Link}}$, for which $\sigma_1 = (\sigma_{\mathsf{ABS}_1}, \sigma_{\mathsf{UCL}_1})$ and $\sigma_2 = (\sigma_{\mathsf{ABS}_2}, \sigma_{\mathsf{UCL}_2})$. He wins if the following are true:

- $\mathsf{Verify}(\sigma_1, m_1, \mathsf{recip}) = \mathsf{Verify}(\sigma_2, m_2, \mathsf{recip}) = 1$
- $\mathsf{Link}(\sigma_1, \sigma_2, \mathsf{recip}) = 0$

The contradiction is straightforward here, non-linkable signatures would lead to $\sigma_{\mathsf{UCL}1} \neq \sigma_{\mathsf{UCL}2}$, where the fact that both signatures verify correctly against the same recipient name recip, would lead to $\sigma_{\mathsf{UCL}1} = \sigma_{\mathsf{UCL}2}$.

In the second case, the adversary aims to break the soundness of the linking algorithm Link by producing supposedly non-linkable signatures (σ_1, σ_2) and yet Link tells that they are linkable. This case can be easily reduced to breaking the linkability property of the LIT scheme, as this can only be done by having $(\mathsf{sk}_1, \mathsf{recip}_1) \neq (\mathsf{sk}_2, \mathsf{recip}_2)$. □

5.7 A Practical DTABS

We now show how to use the construction of ABS-UCL in the ROM to instantiate DTABS in the ROM. The first two parts of the construction of ABS-UCL (see section 5.5) will stay the same. We just need to replace LIT by an IND-CCA encryption scheme. A good candidate for it would be the Cramer-Shoup scheme [CS98] (secure under the DDH assumption) as presented in Fig. 5-1.

Traceability The signer needs to encrypt his identity id, and send it as a part of the signature.

- $C = \mathsf{PKE.Enc}(\mathsf{pk}, \underline{\mathsf{id}}) = (C_1, C_2, C_3, V)$.
- Choose $\beta_{\rho_{\mathsf{id}}}, \beta_r \leftarrow \mathbb{Z}_p$, and compute:
- $\mathcal{C}_1 = {L_1}^{\beta_r}, \mathcal{C}_2 = {L_2}^{\beta_r}, v = \check{\mathcal{H}}(C_1||C_2||C_3)$
- $\mathcal{C}' = K^{-\beta_r} \cdot k_2^{\beta_{\rho_{\mathsf{id}}}}$
- $\mathcal{V} = F^{\beta_r} \cdot (H^v)^{\beta_r}$

$\mathsf{PKE.KeyGen}(1^\lambda)$

- $(\mathbb{G}, p) \leftarrow \mathsf{GrpSetup}(1^\lambda)$.
- $L_1, L_2 \leftarrow \mathbb{G}$; $k, f_1, f_2, h_1, h_2 \leftarrow \mathbb{Z}_p$.
- $F = {L_1}^{f_1} \cdot {L_2}^{f_2}$, $H = {L_1}^{h_1} \cdot {L_2}^{h_2}$, $K = {L_1}^{k}$.
- $\check{\mathcal{H}} : \{0,1\}^* \to \mathbb{Z}_p$
- $\mathsf{pk} = (L_1, L_2, F, H, K, \check{\mathcal{H}})$
- $\mathsf{sk} = (k, f_1, f_2, h_1, h_2)$.

$\mathsf{PKE.Enc}(\mathsf{pk}, M)$

- $r \leftarrow \mathbb{Z}_p$.
- $C_1 = {L_1}^r, C_2 = {L_2}^r, C_3 = K^r \cdot M$
- $v = \check{\mathcal{H}}(C_1||C_2||C_3), V = F^r \cdot H^{rv}$
- Return $C = (C_1, C_2, C_3, V)$.

$\mathsf{PKE.Dec}(\mathsf{sk}, C)$

- Parse C as (C_1, C_2, C_3, V)
- Compute $v = \check{\mathcal{H}}(C_1||C_2||C_3)$
- Check if

$$ {C_1}^{f_1} \cdot {C_2}^{f_2} \cdot ({C_1}^{h_1} \cdot {C_2}^{h_2})^v = V $$

- Return $M = C_3 / {C_1}^{k}$.

Figure 5-1: IND-CCA scheme using Cramer–Shoup

Theses values need to be added to the hash computation, i.e.

$$c = \mathcal{H}_{\mathsf{FS}}(\lambda_j||\mathcal{S}_i||T_i||K_i||U||\hat{K}_i||\hat{U}||\mathcal{B}_j||Z||\mathcal{C}_1||\mathcal{C}_2||\mathcal{C}'||\mathcal{V}), \forall i \in [1,\alpha], \forall j \in [1,\theta].$$

We also need to compute the response related to the random value r, i.e. $\mathfrak{r}_r = \beta_r + c \cdot r$

Verification In the verification, we need to add the parts related to the encryption scheme as follows:

Given $C = (C_1, C_2, C_3, V)$, $\mathsf{pk} = (L_1, L_2, F, H, K, \check{\mathcal{H}})$, compute:

- $v = \check{\mathcal{H}}(C_1||C_2||C_3)$,
- $\mathcal{C}' = K^{\mathfrak{r}_r} \cdot k_2^{-\mathfrak{r}_{\rho_{\mathsf{id}}}} \cdot (C_3/U)^{-c}$
- $\mathcal{C}_1 = {L_1}^{\mathfrak{r}_r} \cdot C_1^{-c}$,
- $\mathcal{C}_2 = {L_2}^{\mathfrak{r}_r} \cdot C_2^{-c}$,
- $\mathcal{V} = F^{\mathfrak{r}_r} \cdot (H^v)^{\mathfrak{r}_{cs}} \cdot V^{-c}$

Similarly, $\mathcal{C}', \mathcal{C}_1, \mathcal{C}_2$, and $\mathcal{V}$ need to be added to the verification of the hash value c.

$$\hat{c} = \mathcal{H}_{\mathsf{FS}}(\lambda_j||\mathcal{S}_i||T_i||K_i||U||\hat{K}_i||\hat{U}||\mathcal{B}_j||Z||\mathcal{C}_1||\mathcal{C}_2||\mathcal{C}'||\mathcal{V})$$

This concludes the signing and verification algorithms in the DTABS scheme. To verify that the opening was done correctly, here are the details (i.e. NIZK_2 as originally defined in Chapter 4) that the opener needs to send along with the signer's identity id.

$\mathbf{NIZK}_2$. Prove

- For $C = \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{id}) = (C_1, C_2, C_3, V)$, compute:
  $C_k = C_1^k \quad C_{f_1} = C_1^{f_1} \quad C_{f_2} = C_2^{f_2} \quad C_{h_1} = C_1^{h_1} \quad C_{h_2} = C_2^{h_2}$,
- $\beta_k, \beta_{f_1}, \beta_{f_2}, \beta_{h_1}, \beta_{h_2} \leftarrow \mathbb{Z}_p$,
- Compute: $C_1^{\beta_k}, C_1^{\beta_{f_1}}, C_2^{\beta_{f_2}}, C_1^{\beta_{h_2}}, C_2^{\beta_{h_2}}$.
- $c = \mathcal{H}_{\text{FS}}(C_1||C_2||C_3||V||C_1^{\beta_k}||C_1^{\beta_{f_1}}||C_2^{\beta_{f_2}}||C_1^{\beta_{h_2}}||C_2^{\beta_{h_2}})$.
- Response is $\Sigma_2 = \{\mathfrak{r}_k = \beta_k + ck, \mathfrak{r}_{f_1} = \beta_{f_1} + cf_1, \mathfrak{r}_{f_2} = \beta_{f_2} + cf_2, \mathfrak{r}_{h_1} = \beta_{h_1} + ch_1, \mathfrak{r}_{h_2} = \beta_{h_2} + ch_2\}$.

$\pi_{\mathsf{Trace}} = (C_1, C_2, C_3, V, C_k, C_{f_1}, C_{f_2}, C_{h_1}, C_{h_2}, c, \Sigma_2)$

$\mathbf{NIZK}_2$.Verify

- $v = \check{\mathcal{H}}(C_1||C_2||C_3)$.
- Check if: $C_{f_1} \cdot C_{f_2} \cdot C_{h_1}^v \cdot C_{h_2}^v = V$
- $\hat{c} = \mathcal{H}_{\text{FS}}(C_1||C_2||C_3||V||C_k^{-c} \cdot C_1^{\mathfrak{r}_k}||C_{f_1}^{-c} \cdot C_1^{\mathfrak{r}_{f_1}}||C_{f_2}^{-c} \cdot C_2^{\mathfrak{r}_{f_2}}||C_{h_1}^{-c} \cdot C_1^{\mathfrak{r}_{h_1}}||C_{h_2}^{-c} \cdot C_2^{s_{h_2}})$
- Check if: $(c = \hat{c} \bigwedge g_2{}^{\mathsf{id}} = C_3/C_k)$

5.8 Conclusion

In this chapter, we have presented our *Attribute Based Signature with User-Controlled Linkability scheme* (ABS–UCL). We first gave the intuition of the scheme and defined the *User-Controlled Linkability* notion. We then gave the syntax and security

definitions of our ABS–UCL. Moreover, we gave the modules needed to generically construct it. We then gave its generic construction, and one practical instantiation in the asymmetric setting in the random oracle model (ROM). We also gave the proofs of the ABS–UCL theorems. We ended the chapter with a *practical* instantiation of DTABS, but this time in the ROM.

Chapter 6

Attribute Based Signatures with Hidden Expressive Policy

6.1 Introduction

In this chapter, we introduce a new attribute based signature scheme, namely, *attribute based signatures with hidden expressive policy from multilinear maps* ABS-HEP. This is a joint work with Liqun Chen, HP Labs, and a patent based on it has been applied for HP [EKC14]. This scheme *fully* hides both the underlying signing policy and the set of attributes being used from the verifier. We give a generic construction of ABS-HEP along with its security model. The construction is based on the existence of multilinear maps. The security properties hold under the multilinear computational Diffie-Hellman assumption. An important feature is that signatures created by the scheme are of a constant size, i.e. independent of the number of attributes presented in the signing policy. For motivational purpose, we will start by giving the following work scenario:

6.1.1 Scenario

Unlike chapters 4 and 5, where the verifier has a say in the policy, we now consider a case where the verifier's concern is only that the document is validly signed according to the rules of the signers organisation.

Let us take a company as an example, although the proposed solution is suitable for various organizations. Assume that the company has employees in different posi-

tions. According to their positions, the employees are allowed to sign some message on behalf of the company, e.g., the CEO alone can create a signature, a certain number of managers in certain levels can work together to make a signature, and ordinary employees can go through a referendum to generate a signature. We want the identities of the signers, along with the internal hierarchy of the company, to be hidden from a signature verifier, as this is sensitive information of the company. The only information that the verifier could learn from a given signature is that the signature could be generated by the same level following company policies, without knowing anything about the order of these levels and the policies.

Another potential example is the following: a university buying cloud services. The university has a complex internal system, e.g. Research Students can spend up to 50 pounds, but above that need their supervisor's signature etc., all of which is of no concern to the cloud supplier as long as the bill is paid, therefore the signing policy can be kept secret. The signer here could be a finance system (not a human being). The university simply doesn't want to expose the internal accounting system, but at the same time, it doesn't want people to spend money which they don't have.

How different is ABS-HEP?

All the previous Attribute Based Signatures used bilinear maps as a building block to construct their schemes. In our scheme, we use multilinear maps that allow us to overcome some of the limitations caused by bilinear maps, namely, the level of expressiveness of the policy. In our scheme, it is possible to go from boolean formulas/span programs (circuits with fanout=1) to deal with general circuits, and yet avoid the *backtracking attack* (see section 6.2) that could have happened at any OR-gate if one wanted to use bilinear maps. This is all possible thanks to the multilinear maps.

The contribution of ABS-HEP is three-fold:

- *Circuits*: This is the first attribute-based signature scheme that can deal with general circuits as the signing policy.
- *Policy privacy*: The proposed attribute-based signature scheme allows the signing policy to be hidden along with the signing attributes, which enhances even more the signer/signer's organization privacy [1].

[1] In practical terms, the signer's organisation can CHANGE policy without needing to inform the verifier.

- *Constant size signature*: The proposed scheme provides a constant size signature and the verification process is very efficient.

6.2 Backtracking Attack

As described in [GGH+13b], a backtracking attack happens when using bilinear maps to walk through a circuit in previous attribute based encryption/signatures schemes. The idea is that when using bilinear maps we only have one target group $\mathbb{G}_T$, and at each node we get a specific target group element. In fig 6-1, one can easily see

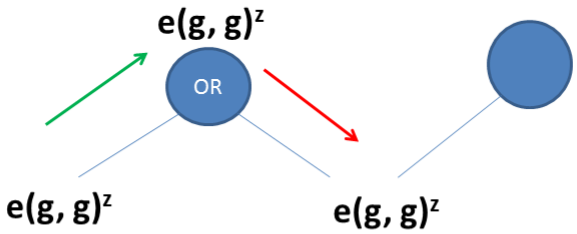


Figure 6-1: ABS for circuits using bilinear maps

how we can reuse an OR gate value to go backward and use it as an input to another gate. In fig 6-2, we can see that different layers of a given circuit have values that belong to different target groups, i.e. $\mathbb{G}_j, \mathbb{G}_{j+1}$, and that is why current constructions of multilinear maps are often referred to as *graded* multilinear maps. Note that in multilinear maps, it is not possible to go back from $\mathbb{G}_{j+1}$ to $\mathbb{G}_j$. Therefore, the OR gate value at layer $j+1$ cannot be used as an input to another gate at the same layer.

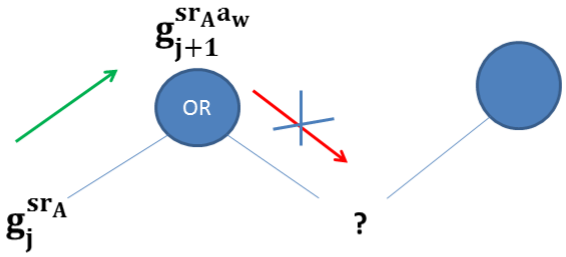


Figure 6-2: ABS for circuits using multilinear maps

6.3 Modules needed to Construct ABS-HEP

Since we are working in the realm where we assume the existence of useful multilinear maps, we will generalize the decision BDH assumption to what we call *k-multilinear assumptions*.

6.3.1 Multilinear maps

Given the groups $\vec{G} = (\mathbb{G}_1, ..., \mathbb{G}_k)$, and the following set:

$$E = \{e_{i,j}; e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \rightarrow \mathbb{G}_{i+j} | i, j \geq 1, i + j \leq k\}$$

We let g_i be a canonical generator of $\mathbb{G}_i$, the map $e_{i,j}$[1] satisfies the following relation;

$$e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab} : \forall a, b \in \mathbb{Z}_p$$

we define the following hardness assumptions related to Multilinear maps:

[1]For ease of exposition, we will often abuse notation and use e instead of $e_{i,j}$.

Definition 10. *k-MDDH:*

The k-Multilinear Decisional Diffie-Hellman (k-MDDH) assumes the following:

Given k groups ($\mathbb{G}_1, \ldots, \mathbb{G}_k$), and generators ($g = g_1, \ldots, g_k$) of prime order p, also $k+1$ group elements in $\mathbb{G}_1$ $\{g^s, g^{c_1}, \cdots, g^{c_k}\}$ for $s, c_1, \ldots, c_k \in \mathbb{Z}_p$.

The probability to distinguish $T = g_k^{s\prod_{j\in[1,k]} c_j}$ from a random group element in $\mathbb{G}_k$ is negligible in the security parameter λ (used initially to generate the groups).

Definition 11. *k-MCDH:*

The k-Multilinear Computational Diffie-Hellman (k-MCDH) assumes the following:

Given k groups ($\mathbb{G}_1, \ldots, \mathbb{G}_k$), and generators ($g = g_1, \ldots, g_k$) of prime order p, also $k+1$ group elements in $\mathbb{G}_1$ $\{g^s, g^{c_1}, \cdots, g^{c_k}\}$ for $s, c_1, \ldots, c_k \in \mathbb{Z}_p$.

Compute $T = g_k^{s\prod_{j\in[1,k]} c_j}$ in $\mathbb{G}_k$.

Observation 1. *Note that $l-$*MDDH *is easy for all $l < k$, for instance, let $l = k-1$, deciding whether T is a random group element in $\mathbb{G}_{k-1}$ or $T = g_{k-1}^{s\prod_{j\in[1,k-1]} c_j}$ can be easily done via the following test:*

$$e(T, g) = e(g_{k-1}^{\prod_{j\in[1,k-1]} c_j}, g^s)$$

where $g_{k-1}^{\prod_{j\in[1,k-1]} c_j}$ is always computable provided the (k-1)-multilinear groups ($\mathbb{G}_1, \ldots, \mathbb{G}_{k-1}$). Note that the computational version is still assumed to be hard.

6.3.2 Simulatable-extractable Signature of Knowledge

We recall the simulatable- extractable signature of knowledge (SimExt-SoK) as it was defined in [CL06]. Let $\mathcal{L}$ be a $\mathcal{NP}$ language, defined by R, a polynomial-time computable binary relation. We define $\mathcal{L}_{\mathsf{R}} = \{\text{statement } y \mid \exists \text{ witness } w : (y, w) \in \mathsf{R}\}$. A *signature of knowledge* (SoK) for a language $\mathcal{L}_{\mathsf{R}}$ consists of the three following algorithms:

- $\mathsf{SoK.Setup}(1^\lambda)$: takes the security parameter and outputs the $\mathcal{P}_{\mathsf{SoK}}$
- $\mathsf{SoK.Sign}(\mathcal{P}_{\mathsf{SoK}}, \mathsf{R}, y, w, m)$: if $(y, w) \in \mathsf{R}$, then outputs a signature σ_{SoK}, otherwise outputs $\perp$.
- $\mathsf{SoK.Verify}(\mathcal{P}_{\mathsf{SoK}}, y, m, \sigma_{\mathsf{SoK}})$: outputs 1 if σ_{SoK} is a valid signature, 0 otherwise.

A signature of knowledge SoK has two security properties. The first is *simulatablity*, which means the existence of an indistinguishable simulator that can sign messages without knowing a witness. The second is *extractablity*, which means the ability to extract a witness from a valid forged signature which wasn't previously queried to the signing oracle. (See 3.8.2 for more details on simulatablity and extractablity.)

6.4 Syntax and Security Definitions of ABS-HEP

6.4.1 Syntax

Let λ be a security parameter, ℓ be the maximum depth of all the circuits describing the policies and n be the maximum input size for all attributes. A circuit f will be defined by its description $(n, q, A, B, \mathsf{GateType})$. The attribute-based signature scheme, ABS-HEP $= (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, includes the following four algorithms.

- ABS-HEP.$\mathsf{Setup}(1^\lambda, n, \ell)$: This outputs the public parameters $\mathcal{P}$ and the master secret key MSK.
- ABS-HEP.$\mathsf{KeyGen}(\mathsf{MSK}, f)$: This takes the master secret key MSK and outputs sk_f.
- ABS-HEP.$\mathsf{Sign}(\mathsf{sk}_f, x, m, f)$: This takes the secret key sk_f and outputs σ if $f(x) = 1$, $\perp$ otherwise.
- ABS-HEP.$\mathsf{Verify}(\sigma, m)$: This outputs 1 if σ verifies correctly against $\mathcal{P}$, 0 otherwise.

6.4.2 Security Definitions

The security of an ABS-HEP scheme includes two notions: *Perfect Privacy* and *Unforgeability*. Note that perfect privacy in the ABS-HEP scheme includes privacy of the policy and privacy of the attributes.

Definition 12. *Perfect Circuit/input privacy: The distribution of signatures on a message m generated via different keys sk_f are indistinguishable, even given the secret keys and master signing key.*

Definition 13. *Unforgeability*

The adversary should not be able to produce a valid ABS signature on a new message m^* even after seeing signatures on different messages of his choice. The unforgeablity game works as follows:

1. *The challenger generates a key pair $(\mathcal{P}, \mathsf{MSK}) \leftarrow \mathsf{ABS.Setup}(1^\lambda, n, l)$ and publishes $\mathcal{P}$.*

2. *The adversary can ask a polynomial number of signatures on messages w.r.t. some circuits f_i by calling the signing oracle $O^{\mathsf{ABS.Sign}(\mathsf{MSK},-)}$ The adversary can ask a polynomial number of signatures on messages w.r.t. to the selected circuit f and selected inputs x^* s.t. $f(x^*) = 1$, by calling the signing oracle $O^{\mathsf{ABS.Sign}(\mathsf{MSK},-)}$.*

3. *Finally, the adversary should output (m^*, σ^*). He wins if;*

   - $\mathsf{ABS\text{-}HEP.Verify}(\sigma^*, m^*) = 1$.
   - *The message m^* has been never queried to the signing oracle.*

The scheme is unforgeable if the advantage of any PPT algorithm $\mathcal{C}$ in the above game is negligible.

6.5 Generic Construction of ABS-HEP

Let λ be a security parameter, ℓ be the maximum depth of all the circuits describing the policies and n be the maximum input size for all attributes. The attribute-based signature scheme, ABS-HEP = (ABS-HEP.Setup, ABS-HEP.KeyGen, ABS-HEP.Sign, ABS-HEP.Verify), includes the following four algorithms.

$\underline{\mathsf{ABS\text{-}HEP.Setup}(1^\lambda, n, \ell)}$

Let $k = \ell + 1$. The setup algorithm takes as input λ, ℓ and n, and produces groups $\vec{G} = (\mathbb{G}_1, ..., \mathbb{G}_{k+1})$ each of prime order $p > 2^\lambda$, with canonical generators $g_1, ..., g_{k+1}$. We let $g = g_1$. Next, it chooses random values $\alpha, \beta \in \mathbb{Z}_p$ and $h_1, ..., h_n \in \mathbb{G}_1$. It sets the master secret key MSK to be

$$(g_{k-1})^\alpha, \beta, h_1, ..., h_n,$$

and outputs the public parameters, pp, which consist of the group sequence description plus

$$K = g_k^{\alpha}, d_0 = g^{\beta}.$$

The algorithm also computes $d_i = h_i^{\beta}$ for each $i \in [1, n]$, which will be used as part of each signing key.

ABS-HEP.KeyGen(MSK, f)

The algorithm takes as input the master secret key and a description f of a circuit, and outputs a secret signing key, sk_f.

To create a specific signing key, the algorithm chooses random values $r_1, ..., r_{n+q} \in \mathbb{Z}_p$, where the random value r_w is associated with wire $w \in [1, n + q]$. The algorithm produces a "header" component

$$K_H = (g_{k-1})^{\alpha - r_{n+q}}.$$

The algorithm then generates key components for every wire w. The structure of the key components depends upon whether w is an input wire, an OR gate or an AND gate, and each case is described as follows.

- *Input wire*

  By convention if $w \in [1, n]$ then it corresponds to the w-th input. The key generation algorithm chooses a random value $z_w \in \mathbb{Z}_p$, and computes the key components for w to be

  $$K_{w,1} = g^{r_w} \cdot h_w^{z_w}, K_{w,2} = g^{-z_w}.$$

- OR *gate*

  Suppose that wire $w \in \mathsf{Gate}$, that $\mathsf{GateType}(w) = \mathsf{OR}$ and that the gate has two inputs called A and B. In addition, let $j = \mathsf{depth}(w)$ be the depth of wire w, and let $r_{A(w)}$ be the r_w value associated with A and $r_{B(w)}$ with B. The algorithm chooses two random values $a_w, b_w \in \mathbb{Z}_p$, and then creates key components for w to be:

  $$K_{w,1} = g^{a_w}, K_{w,2} = g^{b_w}, K_{w,3} = g_j^{r_w - a_w \cdot r_{A(w)}}, K_{w,4} = g_j^{r_w - b_w \cdot r_{B(w)}}.$$

- AND *gate*

  Suppose that wire $w \in \mathsf{Gate}$, that $\mathsf{GateType}(w) = \mathsf{AND}$ and that the gate has two inputs called A and B. In addition, let $j = \mathsf{depth}(w)$ be the depth of wire w, and let $r_{A(w)}$ be the r_w value associated with A and $r_{B(w)}$ with B. The algorithm chooses two random values $a_w, b_w \in \mathbb{Z}_p$, and then creates key components for w to be:

$$K_{w,1} = g^{a_w}, K_{w,2} = g^{b_w}, K_{w,3} = g_j^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}.$$

The algorithm finally outputs the secret signing key, sk_f, including K_H, $d_1, ..., d_n$ together with all the keys for input, OR and AND wires.

Remark 1. *A possible way to implement this scheme and in order to put less trust in signers, one can split the secret key into two parts so the part that contains* $\{\mathsf{sk}_{\mathsf{AND}}, \mathsf{sk}_{\mathsf{OR}}\}$ *will be embedded in a certain signing device. Therefore, singers don't have control over this part of their secret keys.*

ABS-HEP.Sign(sk_f, x, m, f)

Let sk_f be a secret signing key and $x \in \{0,1\}^n$ for which $f(x) = 1$, where x is chosen by the signer. Let $m \in \mathbb{Z}_p$ be a message to be signed. A signature on m under sk_f is a proof that the signer has computed $g_k^{\alpha \cdot \beta}$ without knowing either α or β, and the proof is bound with m. The signature will be created as follows.

First the signer chooses a random value $t \in \mathbb{Z}_p$, compute $c_0 = d_0^t$. We let c_0 be g^s. Note that the value $s = \beta \cdot t$ is unknown to the signer. Now, the signer uses the witness t to produce a signature of knowledge SoK on m, i.e. $\sigma_{\mathsf{SoK}} \leftarrow (\mathcal{P}_{\mathsf{SoK}}, \mathsf{R}, t, m)$.

Next, the signer computes the second part of the final ABS signature denoted by $\sigma_s = g_k^{\alpha \cdot s}$ as follows:

The signer makes a header computation to get

$$E' = e(K_H, c_0) = e(g_{k-1}^{\alpha - r_{n+q}}, g^s) = g_k^{\alpha \cdot s} \cdot g_k^{-r_{n+q} \cdot s}.$$

Then, the signer computes $g_k^{r_{n+q} \cdot s}$. This is done by evaluating the circuit from the bottom up. Consider wire w at depth j; if $f_w(x) = 1$, the signer computes $E_w = (g_{j+1})^{r_w \cdot s}$; if $f_w(x) = 0$, the signer does nothing for that wire. The signer iteratively starts with computing E_1 and proceeds in order to finally compute E_{n+q}.

Computing these values in order ensures that the computation on a depth $j-1$ wire (that evaluates to 1) will be defined before computing for a depth j wire. We show how to compute E_w for all w where $f_w(x) = 1$, again breaking down the cases according to whether the wire is an input, AND or OR gate.

- *Input wire*

  By convention if $w \in [1, n]$ then it corresponds to the w-th input. Suppose that wire w satisfies $f_w(x) = 1$. The algorithm computes $c_w = d_w^t$ and then:

$$E_w = e(K_{w,1}, c_0) \cdot e(K_{w,2}, c_w) = e(g^{r_w} \cdot h_w^{z_w}, g^s) \cdot e(g^{-z_w}, h_w^s) = g_2^{r_w \cdot s}.$$

- OR *gate*

  Consider a wire $w \in \mathsf{Gate}$ and that $\mathsf{GateType}(w) = \mathsf{OR}$. In addition, let $j = \mathsf{depth}(w)$ be the depth of wire w, and $f_w(x) = 1$. We first suppose that $f(Aw) = 1$ and $f(Bw) = 0$, the signer computes:

$$E_w = e(E_{A(w)}, K_{w,1}) \cdot e(K_{w,3}, c_0)$$

$$= e(g_j^{r_{A(w)} \cdot s}, g^{a_w}) \cdot e(g_j^{r_w - a_w \cdot r_{A(w)}}, c_0) = (g_{j+1})^{r_w \cdot s}.$$

  Alternatively, if $f_{A(w)} = 0$, but $f_{B(w)} = 1$, then the signer computes:

$$E_w = e(E_{B(w)}, K_{w,2}) \cdot e(K_{w,4}, c_0)$$

$$= e(g_j^{r_{B(w)} \cdot s}, g^{b_w}) \cdot e(g_j^{r_w - b_w \cdot r_{B(w)}}, c_0) = (g_{j+1})^{r_w \cdot s}.$$

- AND *gate*

  Consider a wire $w \in \mathsf{Gate}$ and that $\mathsf{GateType}(w) = \mathsf{AND}$. In addition, let $j = \mathsf{depth}(w)$ be the depth of wire w. Suppose that $f_w(x) = 1$, i.e., $f_{A(w)}(x) = f_{B(w)}(x) = 1$ and the signer computes:

$$E_w = e(E_{A(w)}, K_{w,1}) \cdot e(E_{B(w)}, K_{w,2}) \cdot e(K_{w,3}, c_0)$$

$$= e(g_j^{r_{A(w)} \cdot s}, g^{a_w}) \cdot e(g_j^{r_{B(w)} \cdot s}, g^{b_w}) \cdot e(g_j^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}, c_0) = (g_{j+1})^{r_w \cdot s}.$$

If $f(x) = f_{n+q}(x) = 1$, the signer gets $E_{n+q} = g_k^{r_{n+q} \cdot s}$, and finally computes $\sigma_s = E' \cdot E_{n+q} = g_k^{\alpha \cdot s}$.

The entire signature is set as

$$\sigma = (c_0, \sigma_{\mathsf{SoK}}, \sigma_s).$$

ABS-HEP.Verify(σ, m)

To verify the candidate signature $\sigma = (c_0, \sigma_{OTP}, \sigma_s)$ on the message m, the verifier does the following two verifications:

- SoK.Verify$(\mathcal{P}_{\mathsf{SoK}}, c_0, m, \sigma_{\mathsf{SoK}})$: by this verification, the verifier is convinced that $c_0 = d_0^t$ and the value t is known by the signer. Recall that $d_0 = g^\beta$ and the signer is not supposed to know the value β.

- Verify σ_s by checking whether the following equation holds.

$$e(\sigma_s, g) = e(K, c_0).$$

  By this verification, the verifier is convinced that $\sigma_s = K^s$ for some value s that is involved in the signer's secret key.

  If both of the verifications pass, the verifier accepts the signature σ; other rejects it. This is because putting these two verifications together, $s = \beta \cdot t$, that is unknown to the signer.

The full proof for the following Theorem is in section 6.6 .

Theorem 12. *The construction of* ABS-HEP *is secure if the k-MCDH assumption holds over multilinear maps, and the signature of knowledge is simulatable-extractable.*

Comparison with ABS for circuits from multilinear maps done in [TLL14]

In [TLL14], and subsequent to our work in [EKC14], they provided an attribute based signature scheme for circuits from multilinear maps. Their scheme shares with ours the fact that it deals with the general type of policies, i.e circuits, and uses multilinear maps as a building block. However, their scheme doesn't achieve the level of privacy that we aim for in our scheme, i.e. hiding the policy and the attributes at the

same time. Their scheme sends the input x to the circuit in the clear with the signatures, because it's needed in the verification process, whereas in our scheme we offer *full privacy* in the sense that the verifier knows nothing about neither of policy and the attributes.

6.6 Proofs

Theorem 13. *The circuit and input, i.e.* (f, x) *of the proposed* ABS-HEP *scheme are perfectly private.*

Proof. Let $\sigma_1 = (c_0, \sigma_{SoK}, \sigma_s) \leftarrow \mathsf{ABS.Sign}(\mathsf{sk}_{f_1}, x^*, m, f_1)$ and $\sigma_2 = (c'_0, \sigma'_{\mathsf{SoK}}, \sigma'_s) \leftarrow \mathsf{ABS.Sign}(\mathsf{sk}_{f_2}, x^*, m, f_2)$. The two distributions are perfectly indistinguishable. This follows directly from the construction and the fact that the signatures are independent of f_i and x_i, $i = 1, 2$. □

Theorem 14. *The construction of* ABS-HEP *is unforgeable if the signature of knowledge* SoK *is simulatable-extracble and the* k*-MCDH assumption holds over the multilinear maps.*

Proof. We will show that if there exists an adversary $\mathcal{C}$ that can break the ABS-HEP scheme, then we can construct an adversary $\mathcal{F}_1$ that can break the simulation-extrability property of the signature of knowledge SoK and another adversary $\mathcal{F}_2$ that can break the k-MCDH assumption.

$$\mathsf{Adv}^{\text{Unforg}}_{\mathsf{ABS},\mathcal{C}}(\lambda) \leq \mathsf{Adv}^{\text{SimExt}}_{\mathsf{SoK},\mathcal{F}_1} + \mathsf{Adv}^{\text{k-MCDH}}_{\mathsf{MultiMaps},\mathcal{F}_2}$$

Adversary $\mathcal{F}_1$:
$\mathcal{F}_2$ sets up the ABS-HEP scheme as in the real case. He picks α, β uniformly at random from $\mathbb{Z}_p$. He publishes the $\mathcal{P}$ as $K = g_k^\alpha$ and $d_0 = g^\beta$. As he is playing against the SimExt property of the SoK, he then has access to its signing oracle. Whenever $\mathcal{C}$ asks for a signing query on (f, m), $\mathcal{F}_2$ picks $t \leftarrow \mathbb{Z}_p$, computes $c_0 = d_0^t$, and sends (t, c_0, d_0, m) to the SoK signing oracle to get σ_{SoK}. He sends the three parts of the ABS-HEP signature back to $\mathcal{C}$. Once he receives the forged signature $(c_0^*, \sigma^*_{\mathsf{SoK}}, \sigma_s^*)$ from $\mathcal{C}$, he sends the second part of the signature, i.e. σ^*_{SoK} to the SoK challenger as the forged signature.

Adversary $\mathcal{F}_2$:
First, $\mathcal{F}_1$ receives the k-MCDH multilinear problem instance i.e, $g, g^s, g^{c_1}, \ldots, g^{c_k}$. $\mathcal{F}_1$ needs to compute $g_k^{s\prod_{i\in[1,k]} c_i}$ in $\mathbb{G}_k$. $\mathcal{F}_1$ can simulate the Setup phase as follows:

- He chooses a random value $\xi \in \mathbb{Z}_p$ and sets $K = g_k^{\alpha} \leftarrow g_k^{\xi+\prod_{i\in[1,k]} c_i}$.
- He sets $\beta = s$ and $d_0 = g^s$,

$\mathcal{F}_1$ outputs the public parameters, pp, including $K = g_k^{\alpha}$ and d_0. $\mathcal{F}_1$ sets up the signature of knowledge by himself, so he can answer signing queries to SoK by running his signing oracle (which doesn't need a witness to sign). Whenever $\mathcal{C}$ asks for a signing query on a tuple (f, m), $\mathcal{F}_1$ first picks a uniformly random $\gamma \leftarrow \mathbb{Z}_p$, computes $\sigma_s = g_k^{\alpha\cdot\gamma}$ and $c_0 = g^{\gamma}$. Now, he runs the simulator of the SoK to sign m without a witness to get σ_{SoK}. The resultant ABS-HEP signature $(c_0, \sigma_{\mathsf{SoK}}, \sigma_s)$ clearly passes both verification tests.

Forgery

- Let σ^* be the forged signature where $\sigma^* = (c_0^*, \sigma_{\mathsf{SoK}}^*, \sigma_s^*)$. The challenger can, with overwhelming probability, extract the secret key $t^* = \log_{d_0} c_0^*$. In addition, the following equation holds:

$$e(\sigma_s^*, g) = e(g_k^{\alpha}, c_0) \tag{6.1}$$

  Notice that RHS of the equation (6.1) is equal to $g_{k+1}^{\alpha\beta t^*}$, which implies that

$$\sigma_s^* = g_k^{\alpha\beta t^*}$$

After the challenger extracted t^*, he can easily compute $(\sigma_s^*)^{1/t^*}$, i.e $g_k^{\alpha\beta}$. Knowing that $g_k^{\alpha} = g_k^{\xi+\prod_{i\in[1,k]} c_i}$, hence;

$$\begin{aligned} g_k^{\alpha\beta} &= g_k^{(\xi+\prod_{i\in[1,k]} c_i)\beta} \\ &= \underbrace{g_k^{\xi\beta}}_{\tau} \cdot g_k^{\beta\prod_{i\in[1,k]} c_i} \end{aligned}$$

τ is computable, simply by lifting g^{β} to g_k^{β} using the multilinear maps, and then raising it to the power of ξ which the challenger knows of, therefore, the challenger's answer

to the k-MCDH challenger is $(\sigma_s^*)^{1/t}/\tau$.

□

6.7 Conclusion

In this chapter, we presented our *Attribute Based Signature with Hidden Expressive Policy scheme* (ABS–HEP). We first gave the intuition of the scheme, the motivation, defined the *hidden expressive policy* notion, and gave a real case scenario. We then gave the syntax and security definitions of our ABS–HEP. Moreover, we gave the modules needed to generically construct it, followed by its generic construction. We concluded this chapter by proving the theorems of ABS–HEP.

Chapter 7

Conclusions and Future Work

In this thesis, we substantially improved the state-of-the-art of Attribute Based Signatures in two important ways. Firstly, we moved Attribute Based Signatures, that use bilinear maps as one of their building blocks, into a *practical* state. In order to do so, we gave two generic solutions to ABS that deal with *Traceability* and *User-Controlled Linkability* respectively. Secondly, we gave a generic construction of another type of ABS where the verifiers don't have a say in the signing policy. Furthermore, the proposed scheme yield a signature that doesn't reveal any information about the signing policy.

The first *enriched* ABS is the DTABS, where we generically added the *Traceability* feature to ABS. The existence of the Traceability is crucial when it comes to putting ABS into practical use. It is a security feature that anonymous schemes like ABS need to have in order to deal with any misuse/abuse cases. A tracing authority can be called on to open any signature subject of dispute. A *Judge of public opinion* makes sure that the tracing authority is saying the truth and no framing is taking place.

The second *enriched* ABS is ABS-UCL, where we generically added the *User-Controlled Linkability* feature to ABS. As traceability is usually thought of as a *trouble-shooting* tool, there appears to be a real need for another tool that can deal with normal use cases of ABS. User-Controlled Linkability offers a way for ABS users to have "active sessions" between themselves, that is actually analogous to the idea of WWW cookies.

Negotiations in general necessitate this type of signature scheme. One practical example of a negotiation is a travel booking system, e.g. holding flights while still looking at hotels, etc. In this case, a given signer doesn't actually need to reveal

anything about his personal identity to convince a certain verifier that he is still talking to the same signer. All what they need to do is to switch on this security feature and send a series of *linkable* signatures, that assure to the verifier that all these signatures are indeed signed by the same signer.

The two important security tools were added to ABS in a modular and generic way, in the sense that, both schemes DTABS and ABS-UCL were proven generically, and the security requirements of the underlying modules were very well defined. Therefore, the efficiency of the concrete constructions would only get better whenever new concrete instantiations of the modules appear. Additionally, all our constructions work in a decentralised fashion where there was no reliance on a central authority at all. Multiple attribute authorities were involved in the schemes, each of them responsible for a certain set of attributes.

Next, we investigated new promising cryptographic tools, e.g. multilinear maps, and we gave the first ABS scheme that *fully* hides both the signing policy and the attributes being used to satisfy it, which offers great levels of anonymity and flexibility. As we mentioned before, a direct application of such a system is related to cloud computing.

This scheme was interesting enough for HP-Labs to protect it and apply for a patent for it. Their main motivation was its potential use in cloud computing (e.g. regulatory compliance).

The proposed ABS scheme deals with the most expressive type of policies, i.e. circuits, and this was not possible without using the multilinear maps tool. We presented a generic construction that relies on two modules, namely the multilinear maps and signature of knowledge. We believe that this work illustrates the great progress that can be made using multilinear maps.

Future Work

There is no doubt that Attribute Based Cryptography is going to play an essential role in the near future, especially with the remarkable growth of the *Cloud Computing* business which has several security requirements to fulfil, including compliance to regulations, etc. Both Attribute Based Encryption and Signatures have their essential and important use in maintaining both confidentiality and authenticity. Therefore, one aims to get even more efficient constructions (in terms of the signature) of *enriched* ABS (i.e. DTABS, ABS-UCL).

So far, all the constructed schemes of attribute based signatures schemes, that deal with policies of predicate type yield signatures of a size that is linear in the number of attributes that appears in the predicate. It would be great if we could come up with an ABS scheme which yields a constant size signature or maybe sub-linear in the number of attributes presented in the signing policy. One possible way to get DTABS and ABS–UCL schemes that yield constant size signature is to deal with less expressive policies. Instead of using signing policies of predicate type, one can use threshold policies, e.g. a given signer should have n out of m attributes presented in the policy to be able to sign. This is an ongoing work with Essam Ghadafi (Bristol University) and Liqun Chen (HP Labs).

The second path is to modularly add both traceability and user-controlled linkability to one ABS scheme. A new security model has to be defined to deal with this scheme. This would give an ABS scheme that can deal with misuse cases and at the same time can give the verifier some control to tell whether or not he is still in the same session and talking with the same anonymous signer.

The fourth path is to use another approach to provide linkability. Instead of giving signatures that can be publicly checked whether or not they are linkable, we can have a *linking authority* that is responsible for checking whether a set of signatures belong to the same anonymous signer or not. The linking authority should not be able to repeal the anonymity of the signatures. A judge is of course needed here to verify if they linking test was done correctly.

The fifth path is to study the applicability of ABS–UCL in the reputation system domain [RKZF00]. Recall that in a reputation system, users can rate products/services which they previously purchased. These systems often have some restrictions so that users are allowed to rate these products only once. Users that try to rate the same

product more than once are misusing the system and therefore their identity should be revealed. A user can anonymously rate different products, he might choose to link his rating, but he can't rate more than once. A tracing authority can be helpful in this case. Further research is needed here to study how compatible existing ABS schemes are with the security requirements of reputations systems.

The sixth path is about the compatibility of multilinear maps with existing schemes, e.g. (NIZK, etc.). As we mentioned earlier, the development and cryptanalysis of multilinear maps are still at nascent stage and some of the recently realised multilinear map schemes are more vulnerable against certain attacks than others. On the other hand, each of these schemes enjoys different "hardness assumptions", i.e. some assumptions that hold for some mulilinear maps construction don't hold for other schemes (e.g. DLIN [BBS04]). The study of the compatibility of existing schemes with the concrete instantiations of the multilinear maps is indeed an intriguing and promising line of research.

References

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of dhies. In *CT-RSA*, pages 143–158, 2001. 166

[ACHO13] M. Abe, S. S. M. Chow, K. Haralambiev, and M. Ohkubo. Double-trapdoor anonymous tags for traceable signatures. In International Journal of Information Security, Springer LNCS 12, Issue 1, pp. 19–31, 2013, 2013. 80

[Adl94] Leonard M. Adleman. The function field sieve. In *Proceedings of the First International Symposium on Algorithmic Number Theory*, ANTS-I, pages 108–121, London, UK, UK, 1994. Springer-Verlag. 41, 46

[AFCK+12] Diego F. Aranha, Laura Fuentes-Castañeda, Edward Knapp, Alfred Menezes, and Francisco Rodríguez-Henríquez. Implementing pairings at the 192-bit security level. *IACR Cryptology ePrint Archive*, 2012:232, 2012. 46

[AGOT14] Masayuki Abe, Jens Groth, Miyako Ohkubo, and Mehdi Tibouchi. Structure-preserving signatures from type ii pairings. Cryptology ePrint Archive, Report 2014/312, 2014. `http://eprint.iacr.org/`. 82, 100

[ALdP11] Nuttapong Attrapadung, Benot Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts, 2011. 26

[AP13] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky thirteen: Breaking the tls and dtls record protocols. In *IEEE Symposium on Security and Privacy*, pages 526–540, 2013. 19

[BB04a] D. Boneh and X. Boyen. Short Signatures Without Random Oracles. In *EUROCRYPT 2004*, Springer LNCS 3027, pp. 56–73, 2004. 82, 86, 115, 166, 169

[BB04b] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-24676-3_14. 25, 166, 170

[BB04c] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Berlin: Springer-Verlag, 2004. Available at `http://www.cs.stanford.edu/~xb/crypto04b/`. 25

[BBD+13] Razvan Barbulescu, Cyril Bouvier, Jérémie Detrey, Pierrick Gaudry, Hamza Jeljeli, Emmanuel Thomé, Marion Videau, and Paul Zimmermann. Discrete logarithm in GF(2^{809}) with FFS. *IACR Cryptology ePrint Archive*, 2013:197, 2013. 33

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *In proceedings of CRYPTO 04, LNCS series*, pages 41–55. Springer-Verlag, 2004. 76, 86, 146, 167

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, October 1988. 55

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 1–15, London, UK, UK, 1996. Springer-Verlag. 18

[BDZ03] Feng Bao, Robert Deng, and HuaFei Zhu. Variations of Diffie-Hellman Problem. 2836:301–312, 2003. 173

[Bei96] Amos Beimel. *Secure schemes for secret sharing and key distribution*. PhD thesis, Technion-Israel Institute of technology, Faculty of computer science, 1996. 87

[Bei11] Amos Beimel. Secret-sharing schemes: A survey. In YeowMeng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology*, volume 6639 of *Lecture Notes in Computer Science*, pages 11–46. Springer Berlin Heidelberg, 2011. 87

[BF01] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO 2001*, pp. 213–229, 2001. 40

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. 24, 42

[BFG13a] D. Bernhard, G. Fuchsbauer, and E. Ghadafi. Efficient Signatures of Knowledge and DAA in the Standard Model. In *ACNS 2013*, Springer LNCS 7954, pp. 518–533, 2013. 113

[BFG+13b] D. Bernhard, G. Fuchsbauer, E. Ghadafi, N.P. Smart, and B. Warinschi. Anonymous attestation with user-controlled linkability. In *International Journal of Information Security*, **12(3)**, pp. 219–249, 2013. 113, 115, 116

[BFI+10] Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch groth-sahai. pages 218–235, 2010. 98, 100

[BFK+06] R. Bobba, O. Fatemieh, F. Khan, C.A. Gunter, and H. Khurana. Using Attribute-Based Access Control to Enable Attribute-Based Messaging. In *ACSAC 2006*, IEEE Computer Society 3027, pp. 403–413, 2006. 27

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 103–112, New York, NY, USA, 1988. ACM. 56

[BKK+09] Joppe W. Bos, Marcelo E. Kaihara, Thorsten Kleinjung, Arjen K. Lenstra, and Peter L. Montgomery. On the security of 1024-bit rsa and 160-bit elliptic curve cryptography. Cryptology ePrint Archive, Report 2009/389, 2009. `http://eprint.iacr.org/`. 33

[BKLS02] Paulo Barreto, Hae Kim, Ben Lynn, and Michael Scott. Efficient algorithms for pairing-based cryptosystems. In Moti Yung, editor, *Advances in Cryptology CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–369. Springer Berlin / Heidelberg, 2002. 48

[BLS03] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. pages 257–267, 2003. 43, 45, 46

[BLS04a] Paulo Barreto, Ben Lynn, and Michael Scott. On the selection of pairing-friendly groups. In Mitsuru Matsui and Robert Zuccherato, editors, *Selected Areas in Cryptography*, volume 3006 of *Lecture Notes in Computer Science*, pages 17–25. Springer Berlin / Heidelberg, 2004. 48

[BLS04b] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In *Journal of Cryptology 2004*, Springer-Verlag, 297-319, 2004. 40, 115

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: formal definition, simplified requirements and a construction based on trapdoor permutations. In Eli Biham, editor, *Advances in cryptology - EUROCRYPT 2003, proceedings of the internarional conference on the theory and application of cryptographic techniques*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629, Warsaw, Poland, May 2003. Springer-Verlag. 21

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *Advances in Cryptology ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer Berlin Heidelberg, 2000. 18

[BN06] Paulo S.L.M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*. 2006. 46, 48

[Bon98] Dan Boneh. The decision diffie-hellman problem. In JoeP. Buhler, editor, *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer Berlin Heidelberg, 1998. 35

[Boy07] X. Boyen. Mesh Signatures. In *EUROCRYPT 2007*, Springer LNCS 4515, pp. 210–227, 2007. 27

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A Paradigm for Designing Efficient Protocols. In *ACM-CCS 1993*, ACM, pp. pp. 62–73, 1993. 8, 23, 28

[BS99] Mihir Bellare and Amit Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In Michael Wiener, editor, *Advances in Cryptology CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 78–78. Springer Berlin / Heidelberg, 1999. 17

[BS02] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. Cryptology ePrint Archive, Report 2002/080, 2002. `http://eprint.iacr.org/`. 67

[BSW06] Dan Boneh, Emily Shen, and Brent Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography - PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 229–240. Springer Berlin Heidelberg, 2006. 20

[BSW07] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption, May 2007. 26

[BWZ14] Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. Cryptology ePrint Archive, Report 2014/930, 2014. `http://eprint.iacr.org/`. 67

[cae12] Caesar: Competition for Autheticated Encryption. Security, Applicability, and Robustness. `http://competitions.cr.yp.to/caesar.html`, 2012. 18

[CH91] David Chaum and Eugene Heyst. Group signatures. In DonaldW. Davies, editor, *Advances in Cryptology EUROCRYPT 91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer Berlin Heidelberg, 1991. 20

[Che06] Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem, 2006. 166

[Che10] L. Chen. A DAA Scheme Requiring Less TPM Resources. In *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 350-365, 2010. 115

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. Cryptology ePrint Archive, Report 2003/182, 2003. `http://eprint.iacr.org/`. 25

[CHL+14] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehle. Cryptanalysis of the multilinear map over the integers. Cryptology ePrint Archive, Report 2014/906, 2014. `http://eprint.iacr.org/`. 67

[CKS08] David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. In Nigel Smart, editor, *Advances in Cryptology EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 127–145. Springer Berlin / Heidelberg, 2008. 36, 167, 172

[CL01] J. Camenisch and A. Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *EUROCRYPT 2001*, Springer LNCS 2045, pp. 93–118, 2001. 27

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *In Advances in Cryptology CRYPTO 06*, pages 78–96. Springer, 2006. 133

[CLT13] Jean-Sbastien Coron, Tancrde Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and JuanA. Garay, ed-

itors, *Advances in Cryptology CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 476–493. Springer Berlin Heidelberg, 2013. 67, 68

[CLT14] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. Cryptology ePrint Archive, Report 2014/975, 2014. `http://eprint.iacr.org/`. 67

[CM09] Sanjit Chatterjee and Alfred Menezes. On cryptographic protocols employing asymmetric pairings - the role of psi revisited. *IACR Cryptology ePrint Archive*, 2009:480, 2009. 25, 42, 43

[CM14] Sanjit Chatterjee and Alfred Menezes. Type 2 structure-preserving signature schemes revisited. Cryptology ePrint Archive, Report 2014/635, 2014. `http://eprint.iacr.org/`. 62, 82, 100

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-45325-3_32. 24

[Cop84] D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *Information Theory, IEEE Transactions on*, 30(4):587 – 594, jul 1984. 44

[CS98] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO 1998*, Springer LNCS 3152, pp. 13–25, 1998. 25, 125

[CS06] Sanjit Chatterjee and Palash Sarkar. On (hierarchical) identity based encryption protocols with short public parameters (with an exposition of waters' artificial abort technique). Cryptology ePrint Archive, Report 2006/279, 2006. `http://eprint.iacr.org/`. 25

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, STOC '91, pages 542–552, 1991. 15, 17

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644 – 654, nov 1976. 13, 19, 34

[EHM11] A. Escala, J. Herranz, and P. Morillo. Revocable Attribute-Based Signatures with Adaptive Security in the Standard Model. In *AFRICACRYPT 2011*, Springer LNCS 6737, pp. 224–241, 2011. 28, 29, 77, 78

[EKC14] Ali El Kaafarani and Liqun Chen. Attribute based signatures with hidden expressive policy. Patent application for Hewlet Packard, PCT/US2014/047773 , filed on 23/07/2014. 2014. 1, 129, 139

[EKCGD14] Ali El Kaafarani, Liqun Chen, Essam Ghadafi, and James Davenport. Attribute-based signatures with user controlled linkability. In *Cryptology and Network Security CANS 2014*, Lecture Notes in Computer Science. Springer International Publishing, 2014. 1, 109

[EKGK13] Ali El Kaafarani, Essam Ghadafi, and Dalia Khader. Decentralized traceable attribute-based signatures. Cryptology ePrint Archive, Report 2013/828, 2013. `http://eprint.iacr.org/`. 1, 70, 116

[EKGK14] Ali El Kaafarani, Essam Ghadafi, and Dalia Khader. Decentralized traceable attribute-based signatures. In Josh Benaloh, editor, *Topics in Cryptology CT-RSA 2014*, volume 8366 of *Lecture Notes in Computer Science*, pages 327–348. Springer International Publishing, 2014. 1, 4, 70, 100

[Eni14] Enisa 2014: Algorithms, Key sizes, and Parameters. `http://www.enisa.europa.eu/`, 2014. 33

[FKMV12] Sebastian Faust, Markulf Kohlweiss, GiorgiaAzzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. In Steven Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012*, volume 7668 of *Lecture Notes in Computer Science*, pages 60–79. Springer Berlin Heidelberg, 2012. 58

[FLA06] K.B. Frikken, J. Li, and M.J. Atallah. Trust negotiation with hidden credentials, hidden policies, and policy cycles. In *NDSS 2006*, The Internet Society, pp. 157–172, 2006. 27

[FR94] G. Frey and H. Ruck. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. page 62:865874, 1994. 41, 44

[Fre06] David Freeman. Constructing pairing-friendly elliptic curves with embedding degree 10. In Florian Hess, Sebastian Pauli, and Michael Pohst, editors, *Algorithmic Number Theory*, volume 4076 of *Lecture Notes in Computer Science*, pages 452–465. Springer Berlin / Heidelberg, 2006. 46

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. 263, 1987. 58, 116

[FST10] David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves, 2010. 10.1007/s00145-009-9048-z. 43, 44, 45, 46

[Fuc09] Georg Fuchsbauer. Automorphic signatures in bilinear groups and an application to round-optimal blind signatures, 2009. 81, 86

[Fuc11] Georg Fuchsbauer. Commuting signatures and verifiable encryption. In *Advances in Cryptology - EUROCRYPT 2011*, pages 224–245. Springer LNCS 6632, May 2011. 81

[Gal05] David Galindo. Boneh-franklin identity based encryption revisited. In Lus Caires, Giuseppe Italiano, Lus Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 102–102. Springer Berlin / Heidelberg, 2005. 10.1007/11523468_64. 25

[Gal12] Steven D. Galbraith. *Mathematics of public key cryptography*. Cambridge University Press, Cambridge, New York, 2012. 23, 36, 37, 38, 39, 40, 47, 48

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006. 25, 170

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and PhongQ. Nguyen,

editors, *Advances in Cryptology EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 2013. 67

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. 8043:479–499, 2013. 9, 131

[Gha14] E. Ghadafi. Stronger Security Notions for Decentralized Traceable Attribute-Based Signatures and More Efficient Constructions. In *Cryptology ePrint Archive, Report 2014/278*, 2014. 100

[GHS02] P. Gaudry, F. Hess, and N. Smart. Constructive and destructive facets of weil descent on elliptic curves. *Journal of Cryptology*, 15:19–46, 2002. 10.1007/s00145-001-0011-x. 40

[Gir14] Damien Giry. Bluekrypt: Cryptographic Key Length Recommendation. `http://www.keylength.com/`, 2014. 33

[GKR04] Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. Secure hashed diffie-hellman over non-ddh groups. In Christian Cachin and JanL. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 361–381. Springer Berlin Heidelberg, 2004. 35, 170

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270 – 299, 1984. 15

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, April 1988. 19

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. 53, 54, 55

[GMV07] S.D. Galbraith, J.F. McKee, and P.C. Valena. Ordinary abelian varieties having small embedding degree. *Finite Fields and Their Applications*, 13(4):800 – 814, 2007. 46

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *J. ACM*, 38(3):690–728, July 1991. 53

[GMY03] Juan A. Garay, Philip MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. In *Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'03, pages 177–194, Berlin, Heidelberg, 2003. Springer-Verlag. 57, 58

[GNSN12] Ma. Gagné, S. Narayan, and R. Safavi-Naini. Short Pairing-Efficient Threshold-Attribute-Based Signature. In *Pairing 2012*, Springer LNCS 7708, pp. 295–313, 2012. 28

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994. 54

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008. 43

[GPSW06] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In *CCS 2006*, ACM ,pp. 89–98, 2006. 24

[Gro06] J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *ASIACRYPT 2006*, Springer LNCS 4284, pp. 444–459, 2006. 58, 92

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *Advances in Cryptology ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*. 2007. 92

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel Smart, editor, *Advances in Cryptology EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer Berlin Heidelberg, 2008. 58, 59, 62

[GS12] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In SIAM Journal on Computing, volume 41(5), pp. 1193–1232, 2012. 83

[GSW09] Practical zero-knowledge proofs for circuit evaluation. In MatthewG. Parker, editor, *Cryptography and Coding*, volume 5921 of *Lecture Notes in Computer Science*. 2009. 98, 100

[GSW10] Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Groth-sahai proofs revisited. In *Public Key Cryptography*, pages 177–192, 2010. 83, 100

[HLLR12] J. Herranz, F. Laguillaumie, B. Libert, and C. Ráfols. Short Attribute-Based Signatures for Threshold Predicates. In *CT-RSA 2012*, Springer LNCS 7178, pp. 51–67, 2012. 28

[HSV06] F. Hess, N.P. Smart, and F. Vercauteren. The eta pairing revisited. Cryptology ePrint Archive, Report 2006/110, 2006. `http://eprint.iacr.org/`. 41

[ISO13] ISO/IEC. ISO/IEC 20008 Information technology. Security techniques –Anonymous digital signatures (All parts), 2013. 112

[JL02] Antoine Joux and Reynald Lercier. The function field sieve is quite special. In *ANTS*, pages 431–445, 2002. 32

[JN03] Antoine Joux and Kim Nguyen. Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups. *J. Cryptology*, 16(4):239–247, 2003. 35, 170

[Jou00] Antoine Joux. A one round protocol for tripartite diffie-hellman. 1838:385–393, 2000. 10.1007/10722028_23. 41

[Jou13a] Antoine Joux. Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields. pages 177–193, 2013. 33

[Jou13b] Antoine Joux. A new index calculus algorithm with complexity $l(1/4 + o(1))$ in very small characteristic. 2013. `http://eprint.iacr.org/`. 33

[Kak10] S.A. Kakvi. Efficient fully anonymous group signatures based on the groth group signature scheme. Masters thesis, University College London, 2010. `http://www5.rz.rub.de:8032/mam/foc/content/publ/thesis_kakvi10.pdf`, 2010. 85, 98, 100

[KCD09] D. Khader, L. Chen, and J. H. Davenport. Certificate-Free Attribute Authentication. In *Cryptography and Coding: IMACC 2009*, Springer LNCS 5921, pp. 301–325, 2009. 28

[KGH83] E.D. Karnin, J. Greene, and M.E. Hellman. On secret sharing systems. *Information Theory, IEEE Transactions on*, 29(1):35–41, Jan 1983. 87

[Kha07] D. Khader. Attribute based group signatures with revocation. In *Cryptology ePrint Archive, Report 2007/241*, `http://eprint.iacr.org/2007/241.pdf`, 2007. 28, 29, 75

[Kil06] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, volume 3876 of *Lecture Notes in Computer Science*, pages 581–600. Springer Berlin Heidelberg, 2006. vii, 84, 85, 96

[Kil07] Eike Kiltz. Chosen-ciphertext secure key-encapsulation based on gap hashed Diffie-Hellman. *IACR Cryptology ePrint Archive*, 2007:36, 2007. 167

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007. 15

[KM05] Neal Koblitz and Alfred Menezes. Pairing-based cryptography at high security levels. In *IMA Int. Conf.*, pages 13–36, 2005. 44

[KM15] Neal Koblitz and Alfred Menezes. The random oracle model: A twenty-year retrospective. Cryptology ePrint Archive, Report 2015/140, 2015. `http://eprint.iacr.org/`. 23

[KSS08] Ezekiel Kachisa, Edward Schaefer, and Michael Scott. Constructing brezing-weng pairing-friendly elliptic curves using elements in the cyclotomic field. In Steven Galbraith and Kenneth Paterson, editors,

Pairing-Based Cryptography Pairing 2008, volume 5209 of *Lecture Notes in Computer Science*, pages 126–135. Springer Berlin / Heidelberg, 2008. 46

[KV08] Eike Kiltz and Yevgeniy Vahlis. CCA2 secure IBE: Standard model efficiency through authenticated symmetric encryption. *IACR Cryptology ePrint Archive*, 2008:20, 2008. 170, 172

[KW93] M. Karchmer and A. Wigderson. On span programs. In *8th IEEE Structure in Complexity Theory*, pp. 102–111, 1993. 86

[KY00] Jonathan Katz and Moti Yung. Complete characterization of security notions for probabilistic private-key encryption. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 245–254, New York, NY, USA, 2000. ACM. 15

[LAS+10] Jin Li, Man Ho Au, Willy Susilo, Dongqing Xie, and Kui Ren. Attribute-based signature and its applications. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 60–69, New York, NY, USA, 2010. ACM. 28

[LK08] J. Li and K. Kim. Attribute-Based Ring Signatures. In *Cryptology ePrint Archive, Report 2008/394*, `http://eprint.iacr.org/2008/394.pdf`, 2008. 28, 75

[LL82] A. K. Lenstra and H. W. Lenstra. Factoring polynomials with rational coefficients. *Muth. Ann*, pages 515–534, 1982. 41

[Men05] Alfred Menezes, editor. *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*. Springer, 2005. 21, 76, 77

[Mil86] Victor S. Miller. Short programs for functions on curves. In *IBM Thomas J. Watson Research Center*, 1986. 47

[MNT01] Miyaji, Nakabayashi, and Takano. New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. *TIEICE: IEICE Transactions on Com-*

munications/Electronics/Information and Systems, 2001. 40, 41, 43, 45, 46

[MOV93] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993. 32, 34, 41, 44, 170

[MPR08] Hemanta Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-Based Signatures: Achieving Attribute-Privacy and Collusion-Resistance. In *Cryptology ePrint Archive, Report 2008/328*, `http://eprint.iacr.org/2008/328.pdf`, 2008. 24, 27, 28, 29

[MPR11] Hemanta Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-Based Signatures. In *CT-RSA 2011*, Springer LNCS 6558, pp. 376–392, 2011. 20, 27, 28, 77, 89, 116

[MRY04] P. MacKenzie, M.K. Reiter, and K. Yang. Alternatives to Non-malleability: Definitions, Constructions, and Applications. In *TCC 2004*, Springer LNCS 2951, pp. 171–190, 2004, 2004. 83

[MZW96] Alfred Menezes, Robert Zuccherato, and Yi-Hong Wu. *An elementary introduction to hyperelliptic curves*. Faculty of Mathematics, University of Waterloo, 1996. 40

[Nac05] David Naccache. Secure and practical identity-based encryption. *IACR Cryptology ePrint Archive*, 2005:369, 2005. 25

[OT11] T. Okamoto and K. Takashima. Efficient Attribute-Based Signatures for Non-Monotone Predicates in the Standard Model. In PKC 2011, Springer LNCS 6571, pp. 35–52, 2011. 28, 29

[OT13] T. Okamoto and K. Takashima. Decentralized Attribute-Based Signatures. In *PKC 2013*, Springer LNCS 7778, pp. 125–142, 2013. 28, 29

[PHP08] Jill Pipher, Jeffrey Hoffstein, and Jill Pipher. *Joseph H. Silverman An Introduction to Mathematical Cryptography*. 2008. 31

[Pol78] J. M. Pollard. Monte carlo methods for index computation mod p. *Mathematics of Computation*, 1978. 32, 43, 46

[PP10] Christof Paar and Jan Pelzl. Message authentication codes (macs). In *Understanding Cryptography*, pages 319–330. Springer Berlin Heidelberg, 2010. 15

[PSV06] D. Page, N. Smart, and F. Vercauteren. A comparison of MNT curves and supersingular curves. *Applicable Algebra in Engineering, Communication and Computing*, 17:379–392, 2006. 10.1007/s00200-006-0017-6. 32, 40, 46

[QQQ+90] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, and Soazig Guillou. How to explain zero-knowledge protocols to your children. In *Advances in CryptologyCRYPTO89 Proceedings*, pages 628–631. Springer, 1990. 53

[RKZF00] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000. 145

[RS92] Charles Rackoff and Daniel Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology CRYPTO 91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer Berlin / Heidelberg, 1992. 15

[RST01] RonaldL. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *Advances in Cryptology ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer Berlin Heidelberg, 2001. 20, 21

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, pages 543–, Washington, DC, USA, 1999. IEEE Computer Society. 57, 88

[SB06] Michael Scott and Paulo Barreto. Generating more mnt elliptic curves, 2006. 10.1007/s10623-005-0538-1. 43

[Sch99] Oliver Schirokauer. Using number fields to compute logarithms in finite fields. *Math. Comp*, 69:1267–1283, 1999. 32

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. 28(4):656–715, October 1949. A footnote on the initial page says: "The material in this paper appeared in a confidential report, 'A Mathematical Theory of Cryptography', dated Sept. 1, 1945 ([**?**]), which has now been declassified.". 14

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO 1984*, pp. 47–53, 1984. 24, 42

[Sma99] N. P. Smart. The discrete logarithm problem on elliptic curves of trace one. *Journal of Cryptology*, 12:193–196, 1999. 10.1007/s001459900052. 32, 40

[Sma13] Nigel Smart. *Cryptography: An Introduction, 3rd Edition*. Mcgraw-Hill College, 2013. 14, 36, 37, 55

[SP92] A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. In *FOCS 1992*, pp. 427–436, 1992. 88

[SSHT12] Naoyuki Shinohara, Takeshi Shimoyama, Takuya Hayashi, and Tsuyoshi Takagi. Key length estimation of pairing-based cryptosystems using η_t pairing. Cryptology ePrint Archive, Report 2012/042, 2012. `http://eprint.iacr.org/`. 41

[SSN09] S. F. Shahandashti and R. Safavi-Naini. Threshold Attribute-Based Signatures and Their Application to Anonymous Credential Systems. In *AFRICACRYPT 2009*, Springer LNCS 5580, pp. 198–216, 2009. 28

[SV07] Nigel P. Smart and Frederik Vercauteren. On computable isomorphisms in efficient asymmetric pairing-based systems. *Discrete Applied Mathematics*, 155(4):538–547, 2007. 43, 49, 50

[SW05] A. Sahai and B. Waters. Fuzzy Identity-Based Encryption. In *EUROCRYPT 2005*, Springer LNCS 3494, pp. 457–473, 2005. 24

[SWD96] Oliver Schirokauer, Damian Weber, and Thomas Denny. Discrete logarithms: The effectiveness of the index calculus method. In Henri Cohen, editor, *Algorithmic Number Theory*, volume 1122 of *Lecture Notes in Computer Science*, pages 337–361. Springer Berlin / Heidelberg, 1996. 10.1007/3-540-61581-4_66. 43

[TLL14] Fei Tang, Hongda Li, and Bei Liang. Attribute-based signatures for circuits from multilinear maps. In ShermanS.M. Chow, Jan Camenisch, LucasC.K. Hui, and SiuMing Yiu, editors, *Information Security*, volume 8783 of *Lecture Notes in Computer Science*, pages 54–71. Springer International Publishing, 2014. 139

[Wat04] Brent R. Waters. Efficient identity-based encryption without random oracles. Cryptology ePrint Archive, Report 2004/180, 2004. `http://eprint.iacr.org/`. 25

[ZKP14] Interactive zero knowledge 3-colorability demonstration., 2014. `http://web.mit.edu/~ezyang/Public/graph/svg.html`. 53

Appendix A

Variants of Diffie–Hellman Problem

- The Computational Diffie–Hellman problem CDH is as follows:

  Given $g, g^a, g^b \in \mathbb{G}$, where $a, b \in \mathbb{Z}_p$.

  Compute g^{ab}.

- The Decisional Diffie–Hellman problem DDH is as follows:

  Given $g, g^a, g^b, g^c \in \mathbb{G}$, where $a, b, c \in \mathbb{Z}_p$.

  Decide if $g^c = g^{ab}$.

- The Generalized Diffie–Hellman problem GDH is as follows;

  Given $g^{\prod_{i \in I} a_i} \in \mathbb{G}$ for all proper subsets $I \subsetneq [k] = \{1, \ldots, k\}$.

  Compute $g^{\prod_{i \in [k]} a_i}$.

- The Square Computational Diffie–Hellman problem SCDH is as follows:

  Given $g, g^x \in \mathbb{G}$.

  Compute g^{x^2}.

- The Inverse Computational Diffie–Hellman problem InvCDH is as follows:

  Given $g, g^x \in \mathbb{G}$.

  Compute $g^{x^{-1}}$.

- The Divisible Computational Diffie–Hellman problem DCDH is as follows:

  Given $g, g^x, g^y \in \mathbb{G}$.

  Compute $g^{\frac{y}{x}}$.

- l–wDH[Che06] or l–DHI[BB04b]

  The l-weak Diffie–Hellman Problem l–wDH is as follows;

  Given $g, g^{\alpha^i} \in \mathbb{G}$ for $i = 1, \ldots, l$.

  Compute $g^{\frac{1}{\alpha}}$.

- q–SDH[BB04a]

  The q–Strong Diffie–Hellman Problem q–SDH is as follows;

  Given $g_1^{\alpha^i} \in \mathbb{G}_1$ for $i = 0, \ldots, l$ and $g_2, g_2^{\alpha} \in \mathbb{G}_2$, a $(l+3)$-tuple of elements

  Compute a pair $(c, g_1^{1/(\alpha+c)}) \in \mathbb{Z}_p \times \mathbb{G}_1$ such that $c \neq 0 \bmod p$. [1]

- HDDH[ABR01]

  The Hash Decisional Diffie–Hellman can be defined as:

  Given a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$, g^u, $g^v \in \mathbb{G}$, where u, $v \in Z_p$, $R \in \{0,1\}^n$ chosen randomly.

  Decide if $R = H(g^{uv})$.

- ODDH[ABR01]

  The Oracle Decisional Diffie–Hellman can be defined as:

  Given $H : \{0,1\}^* \rightarrow \{0,1\}^n$, g^u, $g^v \in G$ where u, $v \in Z_p$. An access to an oracle $\mathcal{H}_v$ defined as $\mathcal{H}_v(X) = H(X^v)$ that can be queried for any element in G except g^u, at the same time, the adversary is allowed to make oracle queries that depend on g^u. $R \in \{0,1\}^n$ chosen randomly.

  Distinguish between $H(g^{uv})$ and R.

- SDH[ABR01] [2]

  The Strong Diffie–Hellman can be defined as:

  Given: g, g^u, g^v, an access to an oracle $\mathcal{O}_v$ defined as $\mathcal{O}_v(U, X)$= 1 if $X = U^v$ and 0 otherwise.

  Compute g^{uv}.

[1] One can easily notice that for a fixed c in the given of q–SDH, and for $\mathbb{G}_1 = \mathbb{G}_2$ then the weaker version of SDH will be equivalent, in terms of hardness, to l–wDH.

[2] Note that this version of Strong Diffie–Hellman, SDH is totally different from the q–SDH.

- GapDH

  The Gap Diffie–Hellman problem states that the computational Diffie–Hellman problem CDH is still hard even though an adversary has access to an oracle that solves DDH.

- GapHDDH[Kil07] [1]

  The Gap Hashed Diffie–Hellman problem is as follows;

  Given $H : \{0,1\}^* \rightarrow \{0,1\}^n$, and a random string $R \in \{0,1\}^n$, and an access to DDH–oracle that can efficiently distinguish between (g^x, g^y, g^{xy}) and (g^x, g^y, g^z).

  Distinguish between $(g^x, g^y, H(g^{xy}))$ and $(g^x, g^y, R))$

- DLIN[BBS04]

  The Decision Linear Diffie–Hellman in a group $\mathbb{G}$ is as follows;

  Given u, v, h and u^a, v^b, h^c

  Decide if $a + b = c$.

- TwinDH[CKS08]

  The Twin Diffie–Hellman is as follows;

  Given $g, X = g^x, Y = g^y$ and $Z = g^z \in \mathbb{G}$.

  Compute g^{xz} and g^{yz}, in other words, compute DH(X, Z) and DH(Y, Z)

- STwinDH

  The Strong Twin Diffie–Hellman is as follows;

  Given: random $X_1, X_2, Y \in G$, an access to a decision oracle that can answer any decisional query of type TwinDH$(X_1, X_2, \hat{Y}) == (\hat{Z}, \hat{W})$, for any input $(\hat{Y}, \hat{Z}, \hat{W}) \in \mathbb{G}^3$.

  Compute: DH(X_1, Y) and DH(X_2, Y).

- TwinDDH

[1] In the literature it's sometimes abbreviated as GHDH.

The Twin decisional Diffie–Hellman problem TwinDDH is to **Distinguish** between the two distributions $(X_1, X_2, Y, \text{DH}(X_1, Y))$ and (X_1, X_2, Y, Z) for random $X_1, X_2, Y, Z \in \mathbb{G}$.

- STwinDDH

  The Strong Twin Decisional Diffie–Hellman problem TwinDD is as follows;

  Given access to a decision oracle that tells for inputs of form $(\hat{Y}, \hat{Z}, \hat{W})$, if $\text{TwinDH}(X_1, X_2, \hat{Y}) = (\hat{Z}, \hat{W})$.

  Distinguish between the two distributions $(X_1, X_2, Y,\ \text{DH}(X_1, Y))$ and (X_1, X_2, Y, Z) for random X_1, X_2, Y, Z, $\in \mathbb{G}$.

  Note that DH(X_2, Y) can not be provided as input to the distinguisher, otherwise the STwinDDH could be broken using the decision oracle.

- STwinHDDH

  The Strong Twin Hashed Decisional Diffie–Hellman problem STwinHDDH is as follows;

  Given a hash function $H : \mathbb{G} \rightarrow \{0,1\}^k$, plus an access to a decision oracle that tells for inputs of form $(\hat{Y}, \hat{Z}, \hat{W})$, if $\text{TwinDH}(X_1, X_2, \hat{Y}) = (\hat{Z}, \hat{W})$.

  Distinguish between the two distributions $(X_1, X_2, Y,\ \text{H}(DH(X_1, Y)))$ and (X_1, X_2, Y, Z) for random X_1, X_2, Y, $Z \in \mathbb{G}$.

  Note that DH(X_2, Y) can not be provided as an input to the distinguisher, otherwise the STwinHDDH could be broken using the decision oracle.

A.1 Variants of Bilinear Diffie-Hellman

We will give more definitions of some other variants of the bilinear Diffie-Hellman.The type of the pairing being used is clear from the context.

- GapBDH

  The Gap Bilinear Diffie–Hellman is as follows;

  Given $X, Y, W \in \mathbb{G}$ and $Z \in \mathbb{G}_T$, and an access to a full decisional oracle that on input $(\hat{X}, \hat{Y}, \hat{W}, \hat{Z})$ checks whether $\text{BDH}(\hat{X}, \hat{Y}, \hat{W}) == \hat{Z}$, for random $\hat{X}, \hat{Y}, \hat{W}, \hat{Z}$

Compute BDH(X, Y, W), i.e. $e(g, g)^{xyw}$ when $X = g^x, Y = g^y, W = g^w$.

- BPI

  The Bilinear Pairing Inversion BPI is as follows;

  Given $g \in \mathbb{G}$ and $z \in \mathbb{G}_T$.

  Compute $h \in \mathbb{G}$ such that $z = e(g, h)$.

- l–BDHI[BB04a]

  The l-Bilinear Diffie–Hellman Inversion l–BDHI is as follows;

  Given g, g^{α^i} for $i = 1, \ldots, l \in \mathbb{G}$.

  Compute $e(g, g)^{\frac{1}{\alpha}}$.

- l–wBDH

  l-weak Bilinear Diffie–Hellman Inversion l–BDHI, for all pairing types, is as follows;

  Given $g_1^{\alpha^i}$ for $i = 0, \ldots, l \in \mathbb{G}_1$ and $g_2, g_2^{\beta} \in \mathbb{G}_2$ for a fixed $l \in \mathbb{N}$.

  Compute $e(g_1, g_2)^{\alpha^{l+1}\beta}$.

- l–BDHE

  l–Bilinear Diffie–Hellman Exponent l–BDHE is as follows;

  Given g, h and g^{α^i} for $i = 1, \ldots, l-1, l+1, \ldots, 2l$

  Compute $e(g, h)^{\alpha^l}$

- TwinBDH

  The Twin Bilinear Diffie–Hellman is as follows;

  Given $X_1, X_2, Y, W \in \mathbb{G}$.

  Compute 2BDH(X_1, X_2, Y, W), i.e. BDH(X_1, Y, W) and BDH(X_2, Y, W).

- STwinBDH

  The Strong Twin Bilinear Diffie–Hellman is as follows;

  Given $X_1, X_2, Y, W \in \mathbb{G}$, and access to a restricted decisional oracle that on input $(\hat{Y}, \hat{W}, \hat{Z}, \hat{T})$ will check if 2BDH$(X_1, X_2, \hat{Y}, \hat{W}) = (\hat{Z}, \hat{T})$.

  Compute BDH(X_1, Y, W) and BDH(X_2, Y, W).

- BDDH[KV08]

  Distinguish between the following distributions:

  $(g, g^x, g^y, g^{y^2}, g^z, e(g,g)^{xyz})$ and $(g, g^x, g^y, g^{y^2}, g^z, e(g,g)^r)$ for $x, y, z, r \in \mathbb{Z}_p$

- q–BDDHI[BB04b]

  Distinguish between the following distributions:

  $(g^x, \ldots, g^{x^q}, e(g,\hat{g})^{\frac{1}{x}})$ and $(g^x, \ldots, g^{x^q}, e(g,g)^r)$, where $q = q(k)$ is a polynomial and $r, x \in \mathbb{Z}_p$

- truncated q–ABDHE[Gen06]

  Distinguish between the following distributions:

  $(g^x, \ldots, g^{x^q}, g^z, g^{zx^{q+2}}, e(g,g)^{zx^{q+1}})$ and $(g^x, \ldots, g^{x^q}, g^z, g^{z.x^{q+2}}, e(g,g)^r)$, where $q = q(k)$ is a polynomial and $r, z, x \in \mathbb{Z}_p$

A.2 Hardness and Reductions

We will start by some facts about the decisional Diffie-Hellman DHH as proven in [JN03]. DDH is easy in any group $\mathbb{G}_1$ accompanied with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$, simply by checking if $e(g^a, g^b) == e(g, g^c)$. If they are equal, then $c = ab$, otherwise c is random. The other case is when we have an asymmetric pairing, i.e. $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. If there exists an isomorphism $\phi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$, which means the pairing is of Type-2, then DDH is easy in $\mathbb{G}_2$. To see that, given $g_1 \in \mathbb{G}_1$ and $g_2, g_2^a, g_2^b, g_2^c \in \mathbb{G}_2$, one can compare $e(g_1, g_2^c)$ to $e(\phi(g_2^a), g_2^b)$, knowing that $g_1 = \phi(g_2)$ is a generator of $\mathbb{G}_1$. On the other hand, DDH in $\mathbb{G}_1$ is still assumed to be hard in Type-2 setting. For Type-3 pairing (When no computable isomorphisms are known between $\mathbb{G}_1$ and $\mathbb{G}_2$), DDH is still considered to be hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$ [1]. Some other known observations about the reductions between some members of the Dlog family of hardness assumptions are listed below.

A.2.1 One way implication

- $\text{Dlog}_{\mathbb{G}_T} \geq \text{Dlog}_{\mathbb{G}_1}$ by [MOV93]

[1] We note here that in some groups where the DDH is known to be easy, the Hashed DDH (HDDH) is still assumed to be hard for a reasonable choice of the hash function H [GKR04]

Given g, $g^a \in \mathbb{G}_1$, compute a.

Let $x \leftarrow e(g,g)$ and $y \leftarrow e(g,g^a)$. It's clear that $y = x^a$; we send them both x and y to the $\text{DLog}_{\mathbb{G}_T}$ oracle to get a.

- DLog$\geq$CDH

  We want to solve CDH using Dlog.

  Given g, g^a, g^b, compute g^{ab}.

  We can simply give g and g^a to the Dlog oracle, the result is a, then we can easily find $(g^b)^a$.

- CDH$\geq$BDH

  Given g, g^a, g^b, g^c, compute $e(g,g)^{abc}$.

  We can first give g^a and g^b to the CDH oracle, the result will be g^{ab}. We can then compute $e(g^{ab}, g^c) = e(g,g)^{abc}$.

- $\text{CDH}_{\mathbb{G}_T} \geq$BDH

  Given g, g^a, g^b, g^c, compute $e(g,g)^{abc}$.

  We let $G = e(g,g) G^a = e(g,g^a), G^b = e(g,g^b)$ and $G^c = e(g,g^c)$. We first give G^a and G^b to the $\text{CDH}_{\mathbb{G}_T}$ oracle to get G^{ab}, and call one more time on input G^{ab} and G^c to G^{abc}.

- CDH$\geq$DDH

  Given g, g^a, g^b, g^c, decide if $g^{ab} = g^c$.

  Using the CDH oracle, we can easily find g^{ab} and then compare it to g^c

- BDH $\geq$DBDH

  Given P, g^a, g^b, g^c, g^z, decide if $e(g,g)^{abc} = e(g,g)^z$.

  We use the CBDH oracle to get $e(g,g)^{abc}$ and then we find $e(g^z, g)$ and we compare the two results.

- $\text{Dlog}_{\mathbb{G}_1} \geq$BDH

  Given g, g^a, g^b, g^c. Find $e(g,g)^{abc}$.

  We can give g, g^a to the Dlog oracle to retrieve a, we then compute $(g^b)^a$, and finally get $r = e(g^{ab}, g^c)$, which is $e(g,g)^{abc}$.

- HDDH$\geq$DDH

  It is easy to notice that if we let the hash function H be the identity function than using the oracle HDDH we can easily recover the DDH, and here lies the importance of HDDH that it is between CDH and DDH.

- SDH$\geq$GapDH

  The reason is that in SDH we have a restricted decisional oracle but in Gap we have the full version of the oracle, where none of its inputs are fixed ahead of querying it.

- DBDH$\geq$BDDH$\geq$2–BDDHI $\geq \ldots \geq q$–BDDHI$\geq$truncated q–ABDHE

  Proof. See [KV08].

  □

A.2.2 Equivalence between some computational variants

- STwinDH $\equiv$ CDH

  It's the main contribution of [CKS08], in which they proved some existing systems secure under the assumptions STwinDH and therefore secure under the standard computational Diffie-Hellman. It is an equivalence between the hardness of two problems where one of them, namely STwinDH, has a *free* decisional oracle offered to the adversary whereas the second is a *pure* computational problem.

  As they mentioned, there is a *trapdoor test* which plays the main role in the proof.

  Trapdoor Test Given a group $\mathbb{G}$ of order q, g is a generator of $\mathbb{G}$. Suppose that $X_1 \in \mathbb{G}$ and $r, s \in \mathbb{Z}_q$ are mutually independent random variables. Let $X_2 = g^s / X_1^r$, and $\hat{Y}, \hat{Z}, \hat{W}$ are random elements in $\mathbb{G}$, each of which is defined as some function of X_1 and X_2, then the following are true;

  – X_2 is uniformly distributed over $\mathbb{G}$;

  – X_1 and X_2 are independent;

– If $X_1 = g^{x_1}$ and $X_2 = g^{x_2}$,
Then the probability that the truth value of

$$\hat{Z}^r \hat{W} = \hat{Y}^s \tag{A.1}$$

doesn't agree with the truth value of

$$\hat{Z} = \hat{Y}^{x_1} \wedge \hat{W} = \hat{Y}^{x_2} \tag{A.2}$$

is at most $1/q$; moreover, if (A.2) holds, then (A.1) certainly holds.

Proof of the equivalence:

– CDH $\geq$STwinDH

It's trivial, since we can just use the CDH oracle twice to solve STwinDH.

– STwinDH $\geq$CDH

Given a challenge instance (X, Y), compute $DH(X, Y)$.

Let's choose r, $s \in Z_q$, set $X_1 = X$ and X_2=g^s/X_1^r. We give the instance (X_1, X_2, Y) to the STwinDH oracle. Suppose that STwinDH makes Q_d queries, of the form, $(\hat{Y}, \hat{Z}_1, \hat{Z}_2)$ to its decisional oracle. Finally, STwinDHwill outputs (Z_1, Z_2). We can test if $Z_1 Z_2^r = Y^s$ holds, then the CDH will output Z_1 as $DH(X, Y)$, otherwise it outputs "failure" with a probability Q_d/q, since the probability of disagreement as stated by the trapdoor test for each query is $1/q$.

- SCDH$\equiv$ CDH

The equivalences between CDH, SCDH, InvCDH, and DCDH have been proved in [BDZ03].

We will prove the equivalence between the four of them as a chain that links them two by two. First, let's start from the equivalence between SCDH and CDH

– SCDH $\geq$CDH

Given g, g^x, g^y, we want to compute g^{xy} using the Oracle SCDH, that acts this way; on input g and g^x the output is g^{x^2}. We will first choose a_1, a_2, b_1, b_2 as auxiliary random values $\in Z_q$. Now let's compute g^{xa_1}, g^{ya_2} and $g^{xa_1b_1+ya_2b_2}$ and give them to the Oracle SCDH. The results will be, $g^{(xa_1)^2}$,$g^{(ya_1)^2}$ and $g^{(xa_1b_1+ya_2b_2)^2}$.Therefore, g^{xy} can be computed.

- CDH $\geq$SCDH

  Given g, g^x we want to compute g^{x^2} using the oracle CDH. we can simply give g, g^x, and g^x to the oracle CDH to get g^{x^2}.

Therefore, we can say that CDH $\equiv$ SCDH.

- SCDH $\equiv$ InvCDH.

  - SCDH $\geq$InvCDH

    Given g, g^x, we want to compute $g^{x^{-1}}$. Recall that the InvCDH oracle works this way; given g, g^x, it outputs g^{x^2}. In other word, given $1st, 2nd = (1st)^n$, it outputs $(1st)^{n^2}$, so if we assign;

    $1st \leftarrow g^x$

    $2nd \leftarrow g$

    Where $2nd = (1st)^{\frac{1}{x}}$, the result will be $(1st)^{(\frac{1}{x})^2} = (g^x)^{\frac{1}{x^2}}$ which is $g^{x^{-1}}$.

  - InvCDH $\geq$SCDH

    Given g, g^x, we want to compute g^{x^2}. We can use the same method for the first implication by defining the oracle as follows; if input is $1st, 2nd = (1st)^n$,then output is $(1st)^{n^{-1}}$, so if we assign;

    $1st \leftarrow g^x$

    $2nd \leftarrow g$

    Where $2nd = (1st)^{\frac{1}{x}}$, we give $1st$ and $2nd$ to the oracle so the result is $(1st)^{(\frac{1}{x})^{-1}} = (g^x)^x$ which is g^{x^2}.

Thus, InvCDH $\iff$ SCDH

- CDH $\equiv$ DCDH

  - CDH $\geq$DCDH

    Given g, g^x, g^y we want to get $g^{\frac{x}{y}}$. Since we are given access to the oracle CDH , then consequently we can have access to the oracle Inv–CDHto get $g^{y^{-1}}$, and finally we give them all g, g^x and $g^{y^{-1}}$ to the oracle CDH in order to get $g^{\frac{x}{y}}$.

– DCDH $\geq$CDH

  Given g, g^x, g^y we want to compute g^{xy}. We choose four auxiliary random values $a_1, b_1, a_2, b_2 \in Z_q$, then we compute (g^{xa_1}, g^{a_2}), and then we feed them to the DCDH oracle to get $u = g^{\frac{xa_1}{a_2}}$. In analogous way we compute (g^{b_1}, g^{yb_2}), and then we feed them to the DCDH oracle to get $v = g^{\frac{b_1}{yb_2}}$. Finally we feed DCDH oracle by both (u, v) to get $g^{\frac{xya_1b_2}{a_2b_1}}$, which immediately gives g^{xy}.

Now we can state that fact the all the above variations of the computational Diffie–Hellamn,

STwinDH$\equiv$ CDH $\equiv$ Inv–CDH$\equiv$ SCDH$\equiv$ DCDH.

Appendix B

Proof of Theorem 4

Lemma 4. *If the* NIZK_1 *and* NIZK_2 *proof systems are zero-knowledge, the tag-based encryption scheme* TPKE *is selective-tag weakly IND-CCA secure, the one-time signature* OTS *is strongly existentially unforgeable, and the hash functions* $\hat{\mathcal{H}}$ *and* $\mathcal{H}$ *are collision-resistant then the construction is fully anonymous (against full-key exposure).*

Proof. We show that if there exists an adversary $\mathcal{B}$ which breaks the anonymity of the construction, we can construct adversaries $\mathcal{F}_1$ against the collision-resistance of the hash function $\hat{\mathcal{H}}$, $\mathcal{F}_2$ against the strong unforgeability of the one-time signature OTS, $\mathcal{F}_3$ against the collision-resistance of the hash function $\mathcal{H}$, $\mathcal{F}_4$ against the NIZK property of the proof system NIZK_1, $\mathcal{F}_5$ against the NIZK property of the proof system NIZK_2, and $\mathcal{F}_6$ against the selective-tag weakly IND-CCA security of the tag-based encryption scheme TPKE.

By the collision-resistance of the hash function $\hat{\mathcal{H}}$, $\mathcal{B}$ has a negligible probability in finding otsvk' such that $\hat{\mathcal{H}}(\mathsf{otsvk}')$ collides with the tag $\hat{\mathcal{H}}(\mathsf{otsvk}^*)$ we will use for the tag-based ciphertext within the challenge signature. If this is not the case, then we can use $\mathcal{B}$ to construct an adversary $\mathcal{F}_1$ that breaks the collision-resistance of $\hat{\mathcal{H}}$.

By the strong existential unforgeability of OTS, we also have that $\mathcal{B}$ has a negligible probability in forging a one-time signature under otsvk^* used in the challenge signature. If this is not the case, we can construct an adversary $\mathcal{F}_2$ that wins the strong unforgeability game of the one-time signature.

By the collision-resistance of the $\mathcal{H}$, $\mathcal{B}$ has a negligible advantage in finding pairs $(\Psi^*, m^*) \neq (\Psi, m)$ such that $\mathcal{H}(\Psi^*, m^*, C_{\text{tbe}}, \hat{\mathcal{H}}(\mathsf{otsvk})) = \mathcal{H}(\Psi, m, C_{\text{tbe}}, \hat{\mathcal{H}}(\mathsf{otsvk}))$. If this is not the case, we can use $\mathcal{B}$ to construct an adversary $\mathcal{F}_3$ which breaks the

collision-resistance of the hash function $\mathcal{H}$. Thus, from now on we assume that there are no such hash collisions.

We now start NIZK_1 in the simulation setting which is by the security of NIZK_1 is indistinguishable from the soundness setting. The proof π is thus now zero-knowledge and hence does not reveal any information about the witness.

We now also start NIZK_2 in the simulation setting which is indistinguishable from the soundness setting. The proof π_{Trace} is now also zero-knowledge and hence $\mathcal{B}$ cannot tell simulated proofs from real proofs.

We now proceed to show how to use $\mathcal{B}$ to construct an adversary $\mathcal{F}_6$ against the selective-tag weakly IND-CCA security of TPKE.

Adversary $\mathcal{F}_6$ runs the Setup algorithm where it starts by randomly choosing a key pair $(\mathsf{otsvk}^*, \mathsf{otssk}^*)$ for OTS that it will use in answering $\mathcal{B}$'s challenge signature. Note that we needed to choose the key pair beforehand as the tag-based encryption scheme is only selective-tag secure and hence the challenger in the ST-WIND-CCA game needs to know the challenge tag before it sends the public-key epk for TPKE. $\mathcal{F}_6$ sends $\hat{\mathcal{H}}(\mathsf{otsvk}^*)$ to its challenger and gets back epk. In its game, $\mathcal{F}_6$ has access to a decryption oracle Dec which it can query on any ciphertext under any tag different from $\hat{\mathcal{H}}(\mathsf{otsvk}^*)$. $\mathcal{F}_6$ chooses crs_1 and crs_2 as simulation reference strings. $\mathcal{F}_6$ also chooses a key pair $(\mathsf{tvk}, \mathsf{tsk})$ for the digital signature scheme DS. $\mathcal{F}_6$ forwards $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathsf{tvk}, \mathbb{A}, \mathcal{H}, \hat{\mathcal{H}})$ to $\mathcal{B}$.

When asked AddA queries, $\mathcal{F}_6$ chooses the secret/verification keys for the authorities itself. Thus, $\mathcal{F}_6$ can answer any AddS queries itself.

To answer the challenge query $\mathsf{CH}_b((\mathsf{id}_0, \mathcal{A}_0), (\mathsf{id}_1, \mathcal{A}_1), m, \Psi)$, $\mathcal{F}_6$ sends $(\mathsf{id}_0, \mathsf{id}_1)$ as its challenge in its ST-WIND-CCA game and gets a ciphertext under the tag $\hat{\mathcal{H}}(\mathsf{otsvk}^*)$ of either the plaintext id_0 or id_1 which he needs to distinguish. $\mathcal{F}_6$ can now construct the rest of the challenge signature by simulating the proof π and signing the whole thing with otssk^* to obtain σ_{ots}.

To answer Trace queries, $\mathcal{F}_6$ just uses its decryption oracle to get the decryption of C_{tbe} which is part of the signature and then simulates proof π_{Trace}. Note that since we have chosen the challenge tag otsvk^* uniformly at random and since we already eliminated any case where any signature sent to Trace uses the same tag as that we used for the challenge signature, such a query will be accepted by $\mathcal{F}_6$'s decryption oracle because the tag is different from the tag used in the challenge ciphertext. The rest of $\mathcal{B}$'s queries are answered normally as in Fig. 4-1.

Finally, when $\mathcal{B}$ outputs its guess, $\mathcal{F}_6$'s output is that of $\mathcal{B}$ $\square$

Lemma 5. *The construction is fully unforgeable if* NIZK_1 *and* NIZK_2 *proof systems are sound, the hash functions* $\mathcal{H}$ *(used in encoding pseudo-attributes) and* $\hat{\mathcal{H}}$ *are collision-resistant, and the one-time signature* OTS*, the digital signature* DS *and the tagged signature* TS *are all existentially unforgeable.*

Proof. We instantiate both NIZK_1 and NIZK_2 proof systems in the soundness setting and hence the adversary cannot break full unforgeability by faking proofs for a false statement. Thus, we proceed to show that if there exists an adversary that wins the full unforgeability game then we can construct adversaries $\mathcal{F}_1$ against the unforgeability of the tagged signature scheme TS, adversary $\mathcal{F}_2$ against the unforgeability of the digital signature scheme DS, adversary $\mathcal{F}_3$ against the strong unforgeability of the one-time signature scheme OTS, and adversaries $\mathcal{F}_4$ and $\mathcal{F}_5$ against the collision-resistance of the hash functions $\mathcal{H}$ and $\hat{\mathcal{H}}$, respectively, such that

$$\mathsf{Adv}^{\text{F-Unforge}}_{\mathsf{DTABS},\mathcal{B}}(\lambda) \leq \kappa(\lambda)\cdot\mathsf{Adv}^{\text{Unfor}}_{\mathsf{TS},\mathcal{F}_1}(\lambda)+\mathsf{Adv}^{\text{Unfor}}_{\mathsf{DS},\mathcal{F}_2}(\lambda)+\mu(\lambda)\cdot\mathsf{Adv}^{\text{Unfor}}_{\mathsf{OTS},\mathcal{F}_3}(\lambda)+\mathsf{Adv}^{\text{Coll}}_{\mathcal{H},\mathcal{F}_4}(\lambda)$$

$$+\mathsf{Adv}^{\text{Coll}}_{\hat{\mathcal{H}},\mathcal{F}_5}(\lambda),$$

where $\kappa(\lambda)$ and $\mu(\lambda)$ are polynomials in λ representing an upper bound on the number of honest attribute authorities and sign queries, respectively, $\mathcal{B}$ is allowed to make in the game.

By the security of the hash function $\mathcal{H}$, $\mathcal{B}$ has a negligible probability in finding collisions between the encodings of different tuples of signing predicate/message/ciphertext/OTS verification key. If this is not the case, we can use $\mathcal{B}$ to construct an adversary $\mathcal{F}_4$ that breaks the collision-resistance of $\mathcal{H}$.

Similarly, by the collision-resistance of the hash function $\hat{\mathcal{H}}$, $\mathcal{B}$ has a negligible probability in finding two different one-time signature keys $\mathsf{otsvk} \neq \mathsf{otsvk}'$ such that $\hat{\mathcal{H}}(\mathsf{otsvk}) = \hat{\mathcal{H}}(\mathsf{otsvk}')$. If this is not the case, we can use $\mathcal{B}$ to construct an adversary $\mathcal{F}_5$ that breaks the collision-resistance of $\hat{\mathcal{H}}$.

Thus, from now on we assume that there are no hash collisions.

- **Adversay $\mathcal{F}_1$:** Adversary $\mathcal{F}_1$ gets the tagged signature scheme's verification key vk from its game and has access to an oracle Sign that it uses to obtain tagged signatures that verify w.r.t. vk on messages and tags (i.e. identities and attributes) of its choice. Adversary $\mathcal{F}_1$ starts by running $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$,

$(\mathsf{crs}_2, \mathsf{xk}_2) \leftarrow \mathsf{NIZK}_2.\mathsf{Setup}(1^\lambda)$ and choosing $(\mathsf{tsk}, \mathsf{tvk})$ honestly. It also creates the key pairs $(\mathsf{esk}, \mathsf{epk})$ for the tag-based encryption scheme TPKE. It then forwards $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathsf{tvk}, \mathbb{A}, \mathcal{H}, \hat{\mathcal{H}})$ and $\mathsf{tk} = \mathsf{esk}$ to $\mathcal{B}$.

Adversary $\mathcal{F}_1$ randomly chooses $i \leftarrow \{1, \ldots, \kappa(\lambda)\}$ and guesses that $\mathcal{B}$'s forgery will involve forging an attribute managed by the attribute authority i. When asked AddA queries, for all authorities $j \neq i$, $\mathcal{F}_1$ chooses the secret/verification keys for the authority itself. For authority i, it sets its verification key to vk it got from its game (and thus it does not know the corresponding secret key). If in the game, $\mathcal{B}$ issues RevealA query on authority i, $\mathcal{F}_1$ aborts the game.

Whenever $\mathcal{B}$ asks AddS queries, if the user has attributes managed by authority i, it forwards such a query to its Sign oracle; Otherwise, it answers the query itself by using the authorities' secret keys available to it.

When asked for Sign queries on $(\mathsf{id}, \mathcal{A}, m, \Psi)$, $\mathcal{F}_1$ first chooses a fresh key pair $(\mathsf{otsvk}, \mathsf{otssk})$ for the one-time signature OTS and encrypts id using $\hat{\mathcal{H}}(\mathsf{otsvk})$ as a tag. It then uses tsk to generate a signature on the pseudo-attribute. Note that by the witness-indistinguishability of NIZK_1, $\mathcal{B}$ cannot tell whether we constructed the signature by using real attributes that id possesses or using a pseudo-attribute. $\mathcal{F}_1$ forwards the signature to $\mathcal{B}$. The rest of $\mathcal{B}$'s queries are answered normally as in Fig. 4-1.

Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{F}_1$ uses the NIZK_1's extraction key xk_1 to extract the witness and returns the tagged signature on the identity and the attribute if $\mathcal{B}$'s forgery involved forging a tagged signature. Otherwise, it aborts. $\mathcal{F}_1$ also aborts if the forgery does not involve forged attributes managed by authority i that $\mathcal{F}_1$ has guessed. The probability that $\mathcal{F}_1$ guesses the correct authority is $\frac{1}{\kappa(\lambda)}$.

By the existential unforgeability of the tagged signature scheme, the probability of $\mathcal{B}$ winning in this case is negligible.

- **Adversary $\mathcal{F}_2$:** Adversary $\mathcal{F}_2$ gets tvk from its game and has access to an oracle Sign that it uses to obtain signatures that verify w.r.t. tvk on messages of its choice. It runs $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$, $(\mathsf{crs}_2, \mathsf{xk}_2) \leftarrow \mathsf{NIZK}_2.\mathsf{Setup}(1^\lambda)$. It also creates the key pair $(\mathsf{esk}, \mathsf{epk})$ for the tag-based encryption scheme TPKE. It then forwards $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathsf{tvk}, \mathbb{A}, \mathcal{H}, \hat{\mathcal{H}})$ and $\mathsf{tk} = \mathsf{esk}$ to $\mathcal{B}$.

When asked for AddA queries, $\mathcal{F}_2$ creates the authority keys itself.

Whenever $\mathcal{B}$ asks AddS queries, $\mathcal{F}_2$ uses the corresponding authorities' secret keys $\mathsf{aask}_{\mathsf{aid}(a)}$ to create the key for the signer.

When asked for Sign queries on $(\mathsf{id}, \mathcal{A}, m, \Psi)$, $\mathcal{F}_2$ generates a fresh key pair $(\mathsf{otsvk}, \mathsf{otssk})$ for the one-time signature. It then produces the tag-based ciphertext C_{tbe} which is the encryption of id using $\hat{\mathcal{H}}(\mathsf{otsvk})$ as a tag and forwards the pseudo-attribute $\mathcal{H}(\Psi, m, C_{\text{tbe}}, \hat{\mathcal{H}}(\mathsf{otsvk}))$ to its oracle. After receiving the signature from its own sign oracle, $\mathcal{F}_2$ produces the proof π using id and the signature on the pseudo-attribute as a witness. It then signs the whole thing with otssk and returns $\sigma = (\sigma_{\mathsf{ots}}, \pi, C_{\text{tbe}}, \mathsf{otsvk})$ as the answer. By the witness-indistinguishability of the proof system NIZK_1, the adversary cannot tell which witness was used in the proof. The rest of $\mathcal{B}$'s queries are answered normally as in Fig. 4-1.

Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{F}_2$ uses NIZK_1's extraction key xk_1 to extract the witness and returns the signature on the pseudo-attribute $a_{m^*, \hat{\Psi}^*, C_{\text{tbe}}, \hat{\mathcal{H}}(\mathsf{otsvk}^*)}$ if the forgery was done by forging a signature on a pseudo-attribute; otherwise, it aborts.

By the existential unforgeability of the of the digital signature scheme DS, the probability of $\mathcal{B}$ winning in this case is negligible.

- **Adversary $\mathcal{F}_3$:** Adversary $\mathcal{F}_3$ gets otsvk from its game and has access to an oracle Sign that it uses to obtain a single one-time signature that verify w.r.t. otsvk on a message of its choice. It runs $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$, $(\mathsf{crs}_2, \mathsf{xk}_2) \leftarrow \mathsf{NIZK}_2.\mathsf{Setup}(1^\lambda)$. It also creates the key pair $(\mathsf{esk}, \mathsf{epk})$ for the tag-based encryption scheme TPKE and $(\mathsf{tsk}, \mathsf{tvk})$ for the digital signature DS. It then forwards $\mathsf{pp} = (\mathsf{crs}_1, \mathsf{crs}_2, \mathsf{epk}, \mathsf{tvk}, \mathbb{A}, \mathcal{H}, \hat{\mathcal{H}})$ and $\mathsf{tk} = \mathsf{esk}$ to $\mathcal{B}$.

  When asked for AddA queries, $\mathcal{F}_3$ creates the authority keys itself. Whenever $\mathcal{B}$ asks AddS queries, $\mathcal{F}_3$ uses the corresponding authorities' secret keys $\mathsf{aask}_{\mathsf{aid}(a)}$ to create the key for the signer.

  Adversary $\mathcal{F}_3$ randomly chooses $i \leftarrow \{1, \ldots, \mu(\lambda)\}$ and guesses that $\mathcal{B}$'s forgery will involve forging a one-time signature that verifies under otsvk used in answering the i-th signing query.

When asked for the j-th Sign query on $(\mathsf{id}, \mathcal{A}, m, \Psi)$, if $j \neq i$, $\mathcal{F}_3$ chooses a fresh key pair $(\mathsf{otsvk}, \mathsf{otssk})$ for the one-time signature scheme and answers the query by itself. If $j = i$, $\mathcal{F}_3$ encrypts id using $\hat{\mathcal{H}}(\mathsf{otsvk})$ (i.e. the public key it got from its game) as a tag to obtain C_{tbe}, it then generates a signature on the pseudo-attribute and constructs the proof π. It then forwards $(\pi, C_{\text{tbe}}, \mathsf{otsvk})$ as the message to its one-time signature signing oracle to get a one-time signature σ_{ots}. $\mathcal{F}_3$ sends the signature $\sigma = (\sigma_{\text{ots}}, \pi, C_{\text{tbe}}, \mathsf{otsvk})$ to $\mathcal{B}$.

The rest of $\mathcal{B}$'s queries are answered normally as in Fig. 4-1.

Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{F}_3$ aborts if the $\mathcal{B}$'s forgery did not involve forging a one-time signature that verifies w.r.t otsvk it got from its game. The probability that $\mathcal{B}$ forges a one-time signature that verifies w.r.t the same otsvk that $\mathcal{F}_3$ has guessed is $\frac{1}{\mu(\lambda)}$.

By the strong existential unforgeability of the one-time signature OTS, $\mathcal{B}$ has a negligible advantage in wining this case.

This concluded the proof.

□

Lemma 6. *The construction is traceable if the* NIZK_1 *proof system is sound, and the digital signature* DS *and the tagged signature* TS *are all existentially unforgeable.*

Proof. Since the NIZK proof systems NIZK_1 is sound, the adversary has a negligible advantage in succeeding by faking proofs for false statements. Thus, we proceed to show that if there exists an adversary that wins the traceability game then we can construct adversaries $\mathcal{F}_1$ attacking the unforgeability of the tagged signature scheme TS, and adversary $\mathcal{F}_2$ attacking the unforgeability of the digital signature scheme DS such that

$$\mathsf{Adv}^{\text{Trace}}_{\mathsf{DTABS},\mathcal{B}}(\lambda) \leq \kappa(\lambda) \cdot \mathsf{Adv}^{\text{Unfor}}_{\mathsf{TS},\mathcal{F}_1}(\lambda) + \mathsf{Adv}^{\text{Unfor}}_{\mathsf{DS},\mathcal{F}_2}(\lambda),$$

where $\kappa(\lambda)$ is a polynomial in λ representing an upper bound on the number of honest attribute authorities $\mathcal{B}$ is allowed to use in the game.

The reductions are very similar to those in the full unforgeability game. The difference here is that $\mathcal{B}$ is not allowed to make any CrptA or RevealA queries.

□

Appendix C

An Example of ABS–HEP

Example 6. ABS-HEP.

Consider the circuit $f = (A \wedge B) \vee C$. *This circuit has* 5 *wires with* 3 *input wires and* 2 *gate (one* OR *and one* AND*), and the* 5*th wire is designated as the output wire. Let* $x \in \{0,1\}^3 = \{A, B, C\}$ *be a possible input string to* f*, we have*

1. $f(\{0,0,0\}) = 0$.
2. $f(\{0,0,1\}) = 1$.
3. $f(\{0,1,0\}) = 0$.
4. $f(\{0,1,1\}) = 1$.
5. $f(\{1,0,0\}) = 0$.
6. $f(\{1,0,1\}) = 1$.
7. $f(\{1,1,0\}) = 1$.
8. $f(\{1,1,1\}) = 1$.

Let λ *be security parameter, the maximum depth of all the circuits be* $\ell = 3$*, the maximum input size for all attributes be* $n = 3$*, the maximum number of the gates is* $q = 2$ *and finally* $k = \ell + 1 = 4$.

The attribute-based signature scheme, ABS-HEP $=$ (ABS.Setup, ABS.KeyGen, ABS-HEP.Sign, ABS-HEP.Verify*), includes the following five algorithms.*

ABS-HEP.Setup$(1^\lambda, n = 3, \ell = 3)$

The setup algorithm produces groups $\vec{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, \mathbb{G}_4)$ *of prime order* p*, with canonical generators* g_1, g_2, g_3, g_4*. We let* $g = g_1$*. Next, it chooses random values* $\alpha, \beta \in \mathbb{Z}_p$ *and* $h_A, h_B, h_C \in \mathbb{G}_1$*, and then computes* $d = g^\beta$ *and* $d_A = h_A^\beta$*,* $d_B = h_B^\beta$ *and* $d_C = h_C^\beta$*. It sets the master secret key* MSK *to be*

$$\mathsf{MSK} = \{g_3^\alpha, \beta, h_A, h_B, h_C\}.$$

It sets $K = g_4^\alpha$*, and outputs the public parameters,* pp*, to be the group sequence description plus*

$$K, d_0.$$

ABS-HEP.KeyGen(MSK, f)

The algorithm takes as input the master secret key and a description f*. The algorithm chooses random values* $r_1, r_2, r_3, r_4, r_5 \in \mathbb{Z}_p$*, where we name* $r_1 = r_A$*,* $r_2 = r_B$*,* $r_3 = r_C$*,* $r_4 = r_{\mathsf{AND}}$*, and* $r_5 = r_{\mathsf{OR}}$*. The algorithm produces a "header" component*

$$K_H = g_3^{\alpha - r_5}.$$

The algorithm then generates key components for every wire (1 to 5). The structure of the key components depends upon whether the wire is an input wire, an OR gate or an AND gate, and each case is described as follows.

- Input wire

  For $w \in \{A, B, C\}$*, the key generation algorithm chooses random values* z_A, z_B*,and* $z_C \in \mathbb{Z}_p$*, and computes the key components as*

  $$K_{A,1} = g^{r_A} \cdot h_A^{z_A}, K_{A,2} = g^{-z_A}.$$

  $$K_{B,1} = g^{r_B} \cdot h_B^{z_B}, K_{B,2} = g^{-z_B}.$$

  $$K_{C,1} = g^{r_C} \cdot h_C^{z_C}, K_{C,2} = g^{-z_C}.$$

- AND gate

  For the wire $w =$ AND*, the gate has two inputs,* A*'s wire and* B*'s wire. In addition, the depth of the wire* AND *is* $\mathsf{depth}(\mathsf{AND}) = 2$*. The algorithm chooses*

random values $a_{\mathsf{AND}}, b_{\mathsf{AND}} \in \mathbb{Z}_p$*, and then creates key components for* AND *to be:*

$$K_{\mathsf{AND},1} = g^{a_{\mathsf{AND}}}, K_{\mathsf{AND},2} = g^{b_{\mathsf{AND}}}, K_{\mathsf{AND},3} = g_2^{r_{\mathsf{AND}} - a_{\mathsf{AND}} \cdot r_A - b_{\mathsf{AND}} \cdot r_B}.$$

- OR gate

  For the wire $w = \mathsf{OR}$*, the gate has two inputs,* AND*'s wire and* C*'s wire. In addition, the depth of wire* OR *is* $\mathsf{depth}(\mathsf{OR}) = 3$*. The algorithm chooses random values* $a_{\mathsf{OR}}, b_{\mathsf{OR}} \in \mathbb{Z}_p$*, and then creates key components for* OR *to be:*

$$K_{\mathsf{OR},1} = g^{a_{\mathsf{OR}}}, K_{\mathsf{OR},2} = g^{b_{\mathsf{OR}}}, K_{\mathsf{OR},3} = g_3^{r_{\mathsf{OR}} - a_{\mathsf{OR}} \cdot r_{\mathsf{AND}}}, K_{\mathsf{OR},4} = g_3^{r_{\mathsf{OR}} - b_{\mathsf{OR}} \cdot r_C}.$$

ABS-HEP.Sign(sk_f, x, m, f)

Let sk_f *be a secret signing key associated with a set of wires in the circuit* f *such that* $f(x) = 1$ *and let* $m \in \mathbb{Z}_p$ *be a message to be signed. A signature on* m *under* sk *is a proof that the signer has computed* $g_4^{\alpha \cdot s}$ *without knowing either* α *or* s*, and the proof is bound with* m*.*

In the Sign *algorithm, the signer chooses a random value* $t \in \mathbb{Z}_p$*, compute* $c_0 = d_0^t$*. We let* $c_0 = g^s$*. Note that the value* $s = \beta \cdot t$ *is unknown to the signer. The signer also computes*

$$c_A = d_A^t = h_A^s, c_B = d_B^t = h_B^s, c_C = d_C^t = h_C^s.$$

Now, the signer uses the witness t *to produce a signature of knowledge* SoK *on* m*, i.e.* $\sigma_{\mathsf{SoK}} \leftarrow (\mathcal{P}_{\mathsf{SoK}}, \mathsf{R}, t, m)$*.*

Next, the signer computes two pieces of knowledge proof denoted by $\sigma_s = g_4^{\alpha \cdot s}$ *and* $E_5 = g_4^{r_5 \cdot s}$ *as follows:*

First, the signer make a header computation to get

$$E' = e(K_H, c) = e(g_3^{\alpha - r_5}, g^s) = g_4^{\alpha \cdot s} \cdot g_4^{-r_5 \cdot s}.$$

Secondly, the signer computes $E_5 = g_4^{r_5 \cdot s}$*.*

Suppose that Signer$_1$ *has* sk_f *including the keys for* A*,* B*,* AND *and* OR*, i.e.,* $x = \{1, 1, 0\}$ *therefore* $f(x) = 1$*. In this case, he will use the following keys:*

$$K_{A,1} = g^{r_A} \cdot h_A^{z_A}, K_{A,2} = g^{-z_A}.$$

$$K_{B,1} = g^{r_B} \cdot h_B^{z_B}, K_{B,2} = g^{-z_B}.$$

$$K_{\mathsf{AND},1} = g^{a_{\mathsf{AND}}}, K_{\mathsf{AND},2} = g^{b_{\mathsf{AND}}}, K_{\mathsf{AND},3} = g_2^{r_{\mathsf{AND}} - a_{\mathsf{AND}} \cdot r_A - b_{\mathsf{AND}} \cdot r_B}.$$

$$K_{\mathsf{OR},1} = g^{a_{\mathsf{OR}}}, K_{\mathsf{OR},3} = g_3^{r_{\mathsf{OR}} - a_{\mathsf{OR}} \cdot r_{\mathsf{AND}}}.$$

Signer$_1$ first computes the two input wires (secret keys are underlined):

$$E_A = e(\underline{K_{A,1}}, c) \cdot e(\underline{K_{A,2}}, c_A) = e(g^{r_A} \cdot h_A^{z_A}, g^s) \cdot e(g^{-z_A}, h_A^s) = g_2^{r_A \cdot s}.$$

$$E_B = e(\underline{K_{B,1}}, c) \cdot e(\underline{K_{B,2}}, c_B) = e(g^{r_B} \cdot h_B^{z_B}, g^s) \cdot e(g^{-z_B}, h_B^s) = g_2^{r_B \cdot s}.$$

Then, he computes the AND *wire:*

$$E_{\mathsf{AND}} = e(E_A, \underline{K_{\mathsf{AND},1}}) \cdot e(E_B, \underline{K_{\mathsf{AND},2}}) \cdot e(\underline{K_{\mathsf{AND},3}}, c)$$

$$= e(g_2^{r_A \cdot s}, g^{a_{\mathsf{AND}}}) \cdot e(g_2^{r_B \cdot s}, g^{b_{\mathsf{AND}}}) \cdot e(g_2^{r_{\mathsf{AND}} - a_{\mathsf{AND}} \cdot r_A - b_{\mathsf{AND}} \cdot r_B}, g^s) = (g_3)^{r_{\mathsf{AND}} \cdot s}.$$

Then, he computes the OR *wire (for $f(\mathsf{AND}) = 1$ and $f(C) = 0$):*

$$E_{\mathsf{OR}} = e(E_{\mathsf{AND}}, \underline{K_{\mathsf{OR},1}}) \cdot e(\underline{K_{\mathsf{OR},3}}, c)$$

$$= e(g_3^{r_{\mathsf{AND}} \cdot s}, g^{a_{\mathsf{OR}}}) \cdot e(g_3^{r_{\mathsf{OR}} - a_{\mathsf{OR}} \cdot r_{and}}, g^s) = g_4^{r_{\mathsf{OR}} \cdot s}.$$

Signer$_1$ has now reached $E_5 = g_4^{r_5 \cdot s}$ as $r_{\mathsf{OR}} = r_5$.

We will show that the signer can achieve the same result if he uses a different x, i.e. $x = \{0, 0, 1\}$, since $f(x) = 1$. Suppose that another signer, say signer$_2$ has sk_f including the keys for C and OR*, i.e.,*

$$K_{C,1} = g^{r_C} \cdot h_C^{z_C}, K_{C,2} = g^{-z_C}.$$

$$K_{\mathsf{OR},2} = g^{b_{\mathsf{OR}}}, K_{\mathsf{OR},4} = g_3^{r_{\mathsf{OR}} - b_{\mathsf{OR}} \cdot r_C}.$$

Signer$_2$ first computes the input wire:

$$E_C = e(\underline{K_{C,1}}, c) \cdot e(\underline{K_{C,2}}, c_C) = e(g^{r_C} \cdot h_C^{z_C}, g^s) \cdot e(g^{-z_C}, h_C^s) = g_2^{r_C \cdot s}.$$

He then lifts the result to the next layer

$$E'_C = e(g_2^{r_C \cdot s}, g) = g_3^{r_C \cdot s}.$$

Then, he computes the OR *wire (for* $f(\mathsf{AND}) = 0$ *and* $f(C) = 1$*):*

$$E_{\mathsf{OR}} = e(E'_C, \underline{K_{\mathsf{OR},2}}) \cdot e(\underline{K_{\mathsf{OR},4}}, c)$$

$$= e(g_3^{r_C \cdot s}, g^{b_{\mathsf{OR}}}) \cdot e(g_3^{r_{\mathsf{OR}} - b_{\mathsf{OR}} \cdot r_C}, g^s) = g_4^{r_{\mathsf{OR}} \cdot s}.$$

Signer$_2$ *has now reached* $E_5 = g_4^{r_5 \cdot s}$.

Clearly, either of the signers can compute $\sigma_s = E' \cdot E_5 = g_4^{\alpha \cdot s}$. *Both* $E_5 = g_4^{r_5 \cdot s}$ *and* σ_s *can be made accessible to the verifier.*

The entire signature is set as

$$\sigma = (c_0, \sigma_{\mathsf{SoK}}, \sigma_s).$$

$\underline{\mathsf{Verify}(\sigma, m)}$

To verify the candidate signature $\sigma = (c_0, \sigma_{\mathsf{SoK}}, \sigma_s)$ *on the message* m*, the verifier does the following verifications:*

- $\mathsf{SoK.Verify}(\mathcal{P}_{\mathsf{SoK}}, c_0, m, \sigma_{\mathsf{SoK}})$*, by following the verification algorithm of the underlying* SoK *signature scheme. By this verification, the verifier is convinced that* $c_0 = d_0^t$ *and the value* t *is known by the signer. Recall that* $d_0 = g^\beta$ *and the signer is not supposed to know the value* β*.*

- *Verify* σ_s *by checking whether the following equation holds.*

  $$e(\sigma_s, g) = e(K, c_0).$$

  By this verification, the verifier is convinced that $\sigma_s = K^s$ *for some value* s *that is involved in the signer's secret key.*